BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC NGUYỄN TẤT THÀNH
KHOA CÔNG NGHỆ THÔNG TIN

TIỂU LUẬN MÔN HỌC

ÁP DỤNG MACHINE LEARNING VÀO PYSPARK ĐỂ DỰ ĐOÁN BỆNH TIỂU ĐƯỜNG

Giảng viên hướng dẫn: ThS. HỒ KHÔI
Sinh viên thực hiện: CHU DOÃN ĐỨC
MSSV: 2000003917
Chuyên ngành: Khoa học dữ liệu
Môn học: Dữ liệu lớn
Khóa: 2020

Tp.HCM, 13 tháng 09 năm 2023

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC NGUYỄN TẤT THÀNH
KHOA CÔNG NGHỆ THÔNG TIN

TIỂU LUẬN MÔN HỌC

ÁP DỤNG MACHINE LEARNING VÀO PYSPARK ĐỂ DỰ ĐOÁN BỆNH TIỂU ĐƯỜNG

Giảng viên hướng dẫn: ThS. HỒ KHÔI
Sinh viên thực hiện: CHU DOÃN ĐỨC
MSSV: 2000003917
Chuyên ngành: Khoa học dữ liệu
Môn học: Dữ liệu lớn
Khóa: 2020

Tp.HCM, 13 tháng 09 năm 2023

BM-ChT-

TRƯỜNG ĐẠI HỌC NGUYỄN TẤT THÀNH
TRUNG TÂM KHẢO THÍ

KỲ THI KẾT THÚC HỌC PHẦN
HỌC KỲ …III… NĂM HỌC …2022… - …2023…

PHIẾU CHẤM THI TIỂU LUẬN/ĐỒ ÁN

Môn thi: Dữ liệu lớn ..... Lớp học phần: 20DTH1D .....

Nhóm sinh viên thực hiện: 1 .....

1. Chu Doãn Đức ..... Tham gia đóng góp: 100%.....
2. ..... Tham gia đóng góp:.....
3. ..... Tham gia đóng góp:.....
4. ..... Tham gia đóng góp:.....
5. ..... Tham gia đóng góp:.....
6. ..... Tham gia đóng góp:.....
7. ..... Tham gia đóng góp:.....
8. ..... Tham gia đóng góp:.....

Ngày thi: 13/09/2023 ..... Phòng thi: L.401 .....

Đề tài tiểu luận/báo cáo của sinh viên: Áp dụng Machine Learning vào PySpark để dự đoán bệnh tiểu đường .....

Phần đánh giá của giảng viên (căn cứ trên thang rubrics của môn học):

Tiêu chí (theo CĐR HP)	**Đánh giá của GV**	**Điểm tối đa**	**Điểm đạt được**
Cấu trúc của báo cáo	....		
Nội dung			
- Các nội dung thành phần	....		
- Lập luận	....		
- Kết luận	.....		
Trình bày	.....		
TỔNG ĐIỂM			

Giảng viên chấm thi
(ký, ghi rõ họ tên)

LỜI CẢM ƠN

Đầu tiên, em xin gửi lời cảm ơn chân thành đến Trường Đại học Nguyễn Tất Thành đã đưa môn học "Dữ liệu lớn" vào trương trình giảng dạy. Đặc biệt, em xin gửi lời cảm ơn sâu sắc đến giảng viên bộ môn – Thầy Hồ Khôi trực tiếp hướng dẫn, dạy dỗ, truyền đạt những kiến thức quý báu cho em trong suốt thời gian học tập vừa qua. Trong thời gian tham gia lớp học của thầy, em đã có thêm cho mình nhiều kiến thức bổ ích, tinh thần học tập hiệu quả, nghiêm túc và đã cho em chắc chắn được hoạch định trong tương lai của mình.

"Dữ liệu lớn" là môn học thú vị, vô cùng bổ ích và có tính thực tế cao. Đảm bảo cung cấp đủ kiến thức, gắn liền với nhu cầu thực tiễn của sinh viên nói chung và riêng bản thân em nói riêng. Tuy nhiên, do vốn kiến thức còn nhiều hạn chế và khả năng tiếp thu thực tế còn nhiều bỡ ngỡ và hạn hẹp. Mặc dù em đã cố gắng hết sức nhưng chắc chắn bài báo cáo của em khó có thể tránh khỏi những thiếu sót và nhiều chỗ còn chưa chính xác, kính mong các thầy/cô chấm bài xem xét và góp ý để bài tiểu luận của em được hoàn thiệt hơn.

Kính chúc thầy có nhiều sức khỏe, hạnh phúc, thành công trên con đường giảng dạy.

Em xin chân thành cảm ơn!

Sinh viên thực hiện

Chu Doãn Đức

LỜI MỞ ĐẦU

Trong thời đại số hóa và phát triển của khoa học công nghệ, việc ứng dụng trí tuệ nhân tạo và học máy đã đóng một vai trò quan trọng trong việc giải quyết những vấn đề y tế phức tạp. Một trong những bệnh thường gặp và có ảnh hưởng lớn đến sức khỏe cộng đồng là bệnh tiểu đường. Bệnh tiểu đường không chỉ gây ra tác động xấu đến chất lượng cuộc sống của người mắc bệnh mà còn tạo ra gánh nặng kinh tế cho hệ thống y tế.

Trong ngữ cảnh này, nghiên cứu này tập trung vào việc áp dụng kỹ thuật Machine Learning trong môi trường PySpark để xây dựng mô hình dự đoán bệnh tiểu đường. PySpark, một thư viện mở rộng của Apache Spark, mang lại sự linh hoạt và hiệu suất trong việc xử lý dữ liệu lớn. Qua việc tận dụng sức mạnh của PySpark, nghiên cứu này mục tiêu xây dựng một mô hình học máy có khả năng dự đoán khả năng mắc bệnh tiểu đường dựa trên các đặc trưng y tế cụ thể.

Bằng cách sử dụng một tập dữ liệu chứa thông tin liên quan đến các yếu tố y tế như đường huyết, huyết áp, BMI và tuổi, chúng tôi sẽ thực hiện các bước tiền xử lý dữ liệu và xây dựng các mô hình học máy mô hình phân loại nhị phân. Sau đó, chúng tôi sẽ đánh giá hiệu suất của các mô hình này thông qua các chỉ số quan trọng như độ chính xác, độ nhạy, độ chính xác dương tính, và F1-score. Nghiên cứu này cũng nhằm mục tiêu thể hiện khả năng ứng dụng của PySpark trong việc xây dựng mô hình dự đoán bệnh tiểu đường trên dữ liệu lớn.

Kết quả của nghiên cứu này có thể đóng góp quý báu vào việc nâng cao khả năng dự đoán bệnh tiểu đường, giúp phát hiện sớm và can thiệp hiệu quả. Đồng thời, việc ứng dụng PySpark trong xây dựng các mô hình học máy sẽ giúp tận dụng tối đa hiệu năng tính toán và khả năng xử lý dữ liệu của môi trường phân tán Apache Spark.

PHẦN NHẬN XÉT + CHẤM ĐIỂM CỦA GIẢNG VIÊN

……
……
……
……
……
……
……
……
……
……
……
……
Điểm giáo viên hướng dẫn: ……
……
Điểm giảng viên chấm vòng 2: ……
……
……
……

TPHCM, Ngày …… tháng …… năm

Giáo viên chấm vòng 2 **Giáo viên hướng dẫn**

MỤC LỤC

DANH MỤC HÌNH

CHƯƠNG 1: GIỚI THIỆU

1.1 –LÝ DO CHỌN ĐỀ TÀI:

Bệnh tiểu đường là một trong những vấn đề sức khỏe quan trọng và phức tạp trên toàn thế giới, ảnh hưởng đến hàng triệu người. Sự gia tăng về tần suất mắc bệnh này đặt ra một thách thức lớn cho hệ thống y tế, đồng thời cần đến những giải pháp sáng tạo để phát hiện và can thiệp kịp thời.

Lý do chọn đề tài "Áp Dụng Machine Learning vào PySpark để Dự Đoán Bệnh Tiểu Đường" là do sự kết hợp mạnh mẽ giữa hai lĩnh vực quan trọng: học máy và xử lý dữ liệu lớn. Dự đoán bệnh tiểu đường thông qua kỹ thuật học máy không chỉ có tiềm năng cải thiện khả năng dự đoán và can thiệp sớm vào bệnh, mà còn mở ra cơ hội nghiên cứu về các yếu tố rủi ro và mối liên hệ tương quan.

PySpark, là một thư viện mở rộng của Apache Spark, đang trở thành một công cụ quan trọng trong xử lý và phân tích dữ liệu lớn. Sự kết hợp giữa PySpark và học máy tạo ra khả năng xử lý dữ liệu lớn và xây dựng các mô hình phức tạp đồng thời. Với khả năng song song và hiệu suất cao của Apache Spark, việc áp dụng machine learning vào PySpark cho phép chúng ta xử lý một lượng lớn dữ liệu y tế và tạo ra các mô hình dự đoán chính xác.

Lý do chọn đề tài này còn nằm ở mong muốn đóng góp thực tiễn cho lĩnh vực y tế. Khả năng dự đoán bệnh tiểu đường giúp tăng cường khả năng phát hiện sớm, cho phép người bệnh và chuyên gia y tế can thiệp kịp thời. Điều này có thể giúp cải thiện chất lượng cuộc sống của những người mắc bệnh và giảm gánh nặng tài chính và y tế đối với xã hội.

Tóm lại, việc kết hợp lĩnh vực học máy và PySpark để dự đoán bệnh tiểu đường không chỉ có ý nghĩa khoa học mà còn mang lại giá trị thực tiễn, đóng góp quan trọng vào việc nâng cao sức khỏe cộng đồng và phát triển y học dự phòng.

1.2 – MỤC TIÊU CỦA ĐỀ TÀI:

Xây dựng một hệ thống dự đoán khả năng mắc bệnh tiểu đường thông qua việc kết hợp sức mạnh của kỹ thuật học máy và khung làm việc PySpark. Qua đó kết hợp sức mạnh của Machine Learning và PySpark để xây dựng mô hình dự đoán bệnh tiểu đường, đồng thời khám phá ưu điểm của việc ứng dụng PySpark trong việc xử lý dữ liệu lớn và tối ưu hóa tính toán. Điều này mang lại giá trị nghiên cứu lý thú và ứng dụng thực tế trong việc cải thiện chất lượng cuộc sống và quản lý y tế.

1.3 – MÔI TRƯỜNG PHÁT TRIỂN:

Python và PySpark:

- Python là ngôn ngữ lập trình chính được sử dụng trong việc xây dựng mô hình và tiền xử lý dữ liệu.
- PySpark là một phần quan trọng trong việc tối ưu hóa việc xử lý dữ liệu lớn, đặc biệt là khi làm việc với Spark DataFrame. Các thư viện Python như pyspark, pandas, numpy, và scikit-learn sẽ được sử dụng để thực hiện các bước trong quá trình nghiên cứu.

Tập Dữ Liệu: Để thực hiện nghiên cứu, cần có tập dữ liệu chứa thông tin y tế liên quan đến các đặc trưng như đường huyết, huyết áp, BMI và tuổi của các cá nhân. Tập dữ liệu này có thể được thu thập từ các nguồn như bệnh viện, trung tâm y tế hoặc cơ sở dữ liệu y tế.

Môi trường phát triển của đề tài này yêu cầu sự kết hợp giữa Python, PySpark và các thư viện học máy, cùng với tập dữ liệu y tế thích hợp. Việc sử dụng môi trường phát triển này sẽ giúp tối ưu hóa quá trình xử lý dữ liệu lớn và xây dựng các mô hình dự đoán.

1.4 – CÔNG CỤ HỖ TRỢ:

- Ngôn ngữ: Python.
- Công cụ xây dựng thuật toán/mô hình: Google Colab.
- Các thư viện của Python: Pandas, Numpy, Sk-learn, PySpark.

CHƯƠNG 2: XÂY DỰNG ỨNG DỤNG THUẬT TOÁN

2.1 - MÔ TẢ BÀI TOÁN:

Mô hình phân loại nhị phân là một loại mô hình học máy được sử dụng để phân loại các đối tượng thành hai nhóm dựa trên một thuộc tính cụ thể. Có nhiều phương pháp thích hợp cho việc học phân loại nhị phân, bao gồm cây quyết định, mạng Bayes, support vector machine và mạng nơron.

Để đánh giá hiệu quả của một mô hình phân loại nhị phân, người ta thường sử dụng các khái niệm độ nhạy và đặc trưng. Độ nhạy (sensitivity) là tỉ lệ của số người bị bệnh được xác định đúng là có bệnh trên tổng số người bị bệnh, nghĩa là (dương tính đúng)/ (dương tính đúng + âm tính sai). Đặc trưng (specificity) là tỉ lệ của số người không bị bệnh có kết quả xét nghiệm âm tính trên tổng số người không có bệnh (thực), nghĩa là (âm tính đúng)/ (âm tính đúng + dương tính sai).

Với một cách xác định một lớp là positive, Precision được định nghĩa là tỉ lệ số điểm true positivetrong số những điểm được phân loại là positive (TP + FP).

Recall được định nghĩa là tỉ lệ số điểm true positive trong số những điểm thực sự là positive (TP + FN).

Một cách toán học, Precison và Recall là hai phân số có tử số bằng nhau nhưng mẫu số khác nhau:

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

- True Positive (TP): Số lượng các điểm dữ liệu thuộc lớp positive (thực tế) mà mô hình đã dự đoán đúng là thuộc lớp positive.
- False Positive (FP): Số lượng các điểm dữ liệu thuộc lớp negative (thực tế) mà mô hình đã dự đoán sai là thuộc lớp positive.

- False Negative (FN): Số lượng các điểm dữ liệu thuộc lớp positive (thực tế) mà mô hình đã dự đoán sai là thuộc lớp negative.

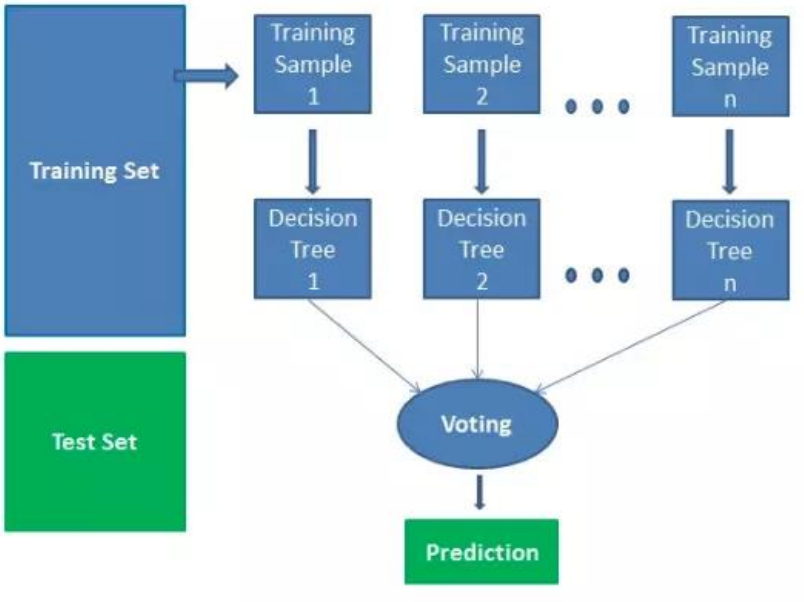
Chính xác (Precision): Chính xác đo lường khả năng của mô hình trong việc dự đoán lớp positive một cách chính xác. Nó đánh giá tỷ lệ giữa số lượng dự đoán positive đúng (TP) so với tổng số dự đoán positive (TP + FP). Precision thường được sử dụng trong các tình huống mà việc tránh các dự đoán sai positive là quan trọng.

Độ Nhạy (Recall): Độ nhạy đo lường khả năng của mô hình trong việc tìm ra tất cả các điểm dữ liệu thuộc lớp positive thực tế. Nó đánh giá tỷ lệ giữa số lượng dự đoán positive đúng (TP) so với tổng số mẫu thực tế positive (TP + FN). Độ nhạy thường được sử dụng trong các tình huống mà việc không bỏ sót bất kỳ điểm dữ liệu positive nào là quan trọng.

Cả Precision và Recall đều cung cấp thông tin quan trọng về khả năng của mô hình phân loại. Tuy nhiên, chúng có mối quan hệ đảo ngược: khi Precision tăng lên, Recall có thể giảm và ngược lại. Do đó, việc cân nhắc và lựa chọn thích hợp giữa Precision và Recall phụ thuộc vào bối cảnh ứng dụng cụ thể và mục tiêu đánh giá.

2.1.1 – Phân loại Rừng ngẫu nhiên (Random Forest Classifier):

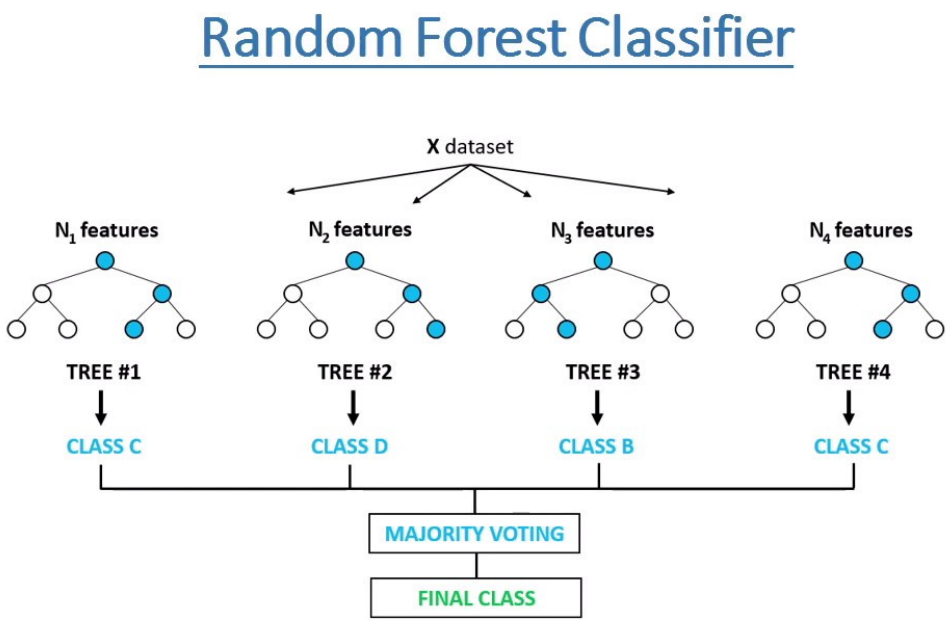
Rừng ngẫu nhiên là một thuật toán học có giám sát được sử dụng cho cả phân loại và hồi quy. Tuy nhiên, nó chủ yếu được sử dụng cho các vấn đề phân loại. Như chúng ta biết, một khu rừng được tạo thành từ nhiều cây và càng nhiều cây thì khu rừng càng mạnh mẽ. Tương tự như vậy, thuật toán rừng ngẫu nhiên tạo ra các cây quyết định trên các mẫu dữ liệu và sau đó lấy dự đoán từ mỗi cây và cuối cùng chọn giải pháp tốt nhất bằng cách bỏ phiếu. Đây là một phương pháp tập hợp tốt hơn so với một cây quyết định đơn lẻ vì nó giảm thiểu việc quá khớp bằng cách lấy trung bình kết quả.



Hình 2. 1: Mô hình Random Forest Classifier

2.1.2 – Phân loại Cây quyết định (Decision Tree Classifier):

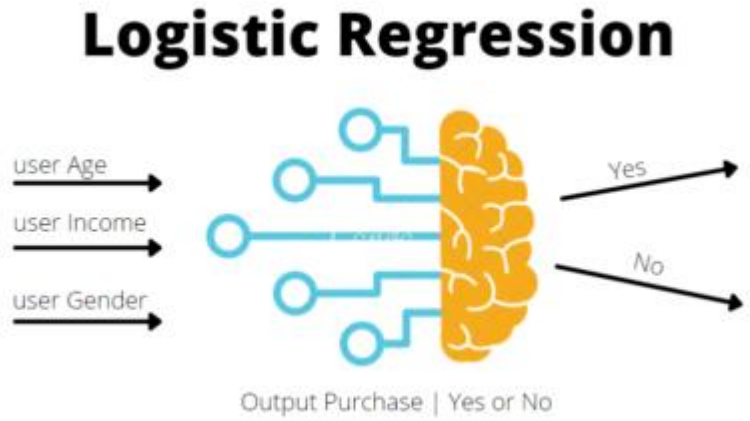
Cây quyết định được sử dụng rộng rãi vì chúng dễ hiểu, xử lý được các đặc trưng phân loại, mở rộng đến cài đặt phân loại đa lớp, không yêu cầu tỷ lệ đặc trưng và có khả năng nắm bắt các tính phi tuyến và tương tác đặc trưng.



Hình 2. 2: Mô hình Decision Tree Classifier

2.1.3 – Hồi quy Logistic (Logistic Regression):

Hồi quy logistic là phân tích hồi quy thích hợp để tiến hành khi biến phụ thuộc là nhị phân (nhị phân). Giống như tất cả các phân tích hồi quy, hồi quy logistic là một phân tích dự đoán. Hồi quy logistic được sử dụng để mô tả dữ liệu và giải thích mối quan hệ giữa một biến nhị phân phụ thuộc và một hoặc nhiều biến độc lập ở cấp độ danh nghĩa, thứ tự, khoảng hoặc tỷ lệ. Hồi quy Logistic được sử dụng khi biến phụ thuộc (mục tiêu) là phân loại.



Hình 2. 3: Mô hình Logistic Regression

2.2 - XÂY DỰNG BỘ DỮ LIỆU:

Theo Tổ chức Y tế Thế giới (WHO), tiểu đường là một bệnh mãn tính xảy ra khi tuyến tụy không sản xuất đủ insulin hoặc khi cơ thể không thể sử dụng hiệu quả insulin mà nó sản xuất. Insulin là một hormone điều chỉnh đường huyết. Tăng đường huyết, hoặc đường huyết tăng cao, là một tác dụng phổ biến của tiểu đường không kiểm soát và theo thời gian dẫn đến tổn thương nghiêm trọng cho nhiều hệ thống của cơ thể, đặc biệt là dây thần kinh và mạch máu.

Từ năm 2000 đến năm 2016, tỷ lệ tử vong sớm (tức là trước tuổi 70) do tiểu đường tăng 5%. Tại các nước thu nhập cao, tỷ lệ tử vong sớm do tiểu đường giảm từ năm 2000 đến

năm 2010 nhưng sau đó tăng trong giai đoạn 2010-2016. Tại các nước thu nhập trung bình thấp, tỷ lệ tử vong sớm do tiểu đường tăng trong cả hai giai đoạn.

Dữ liệu bệnh tiểu đường gồm 9 biến giải thích được thu thập từ trang web chính thức của Kaggle.com. Các biến giải thích là Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age, Outcome.

Bộ dữ liệu bao gồm (Dữ liệu đã được làm sạch):

- Pregnancies: Số lượng thai kỳ.
- Glucose: Đường huyết.
- BloodPressure: Huyết áp.
- SkinThickness: Độ dày của da.
- Insulin: Lượng Insulin.
- BMI: Chỉ số khối cơ thể (Body Mass Index).
- DiabetesPedigreeFunction: Hệ số di truyền bệnh tiểu đường.
- Age: Tuổi.
- Outcome: Kết quả, trong trường hợp này, có bệnh tiểu đường hay không (1 nếu có, 0 nếu không).

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

Hình 2. 4: Dữ liệu bệnh tiểu đường

2.3 – XÂY DỰNG BÀI TOÁN DỰ ĐOÁN:

Xây dựng một mô hình dự đoán khả năng mắc bệnh tiểu đường dựa trên các thông tin y tế của các cá nhân. Mục tiêu là tạo ra một mô hình có khả năng phân loại mỗi cá nhân là có bệnh tiểu đường hoặc không có bệnh tiểu đường dựa trên các đặc trưng được cung cấp.

Cho tập dữ liệu chứa thông tin y tế về các đặc trưng như số lượng thai kỳ, đường huyết, huyết áp, độ dày của da, lượng Insulin, chỉ số khối cơ thể (BMI), hệ số di truyền bệnh tiểu đường và tuổi của các cá nhân. Mỗi cá nhân trong tập dữ liệu có kết quả là có bệnh tiểu đường (1) hoặc không có bệnh tiểu đường (0).

Công việc cần thực hiện:

- Tiền xử lý dữ liệu: Loại bỏ dữ liệu thiếu, chuẩn hóa đặc trưng, chia tập dữ liệu thành tập huấn luyện và tập kiểm tra.
- Xây dựng mô hình: Sử dụng các thuật toán machine learning như Hồi quy Logistic, Random Forest, Decision Tree hoặc Gradient Boosting để xây dựng mô hình phân loại.
- Đánh giá hiệu suất: Đánh giá mô hình bằng các chỉ số như độ chính xác, độ nhạy, độ chính xác dương tính và F1-score.
- Lựa chọn mô hình: So sánh hiệu suất của các mô hình và chọn mô hình tốt nhất cho bài toán.
- Triển khai mô hình: Sử dụng mô hình đã chọn để dự đoán khả năng mắc bệnh tiểu đường cho các cá nhân mới.

CHƯƠNG 3: XÂY DỰNG MÔ HÌNH THỰC NGHIỆM

3.1 – NHẬP THƯ VIỆN VÀ KHÁM PHÁ DỮ LIỆU:

3.1.1 – Nhập thư viện:

Đầu tiên khi sử dụng các thư viện của Python, chúng ta phải cài đặt môi trường PySpark để áp dụng vào đề tài.

- !pip install pyspark: Cài đặt thư viện PySpark bằng lệnh pip. PySpark là một gói Python cho Apache Spark, một hệ thống xử lý dữ liệu phân tán mã nguồn mở.
- Import pandas, numpy, skleanr: Các thư viện liên quan để phân tích dữ liệu, toán học, huấn luyện và đánh giá mô hình.

```
[ ] !pip install pyspark

    Requirement already satisfied: pyspark in /usr/local/lib/python3.10/dist-packages (3.4.1)
    Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.10/dist-packages (from pyspark) (0.10.9.7)

[ ] # import thư viện cần thiết
    import pandas as pd
    import numpy as np
    import sklearn

[ ] from google.colab import drive
    drive.mount('/content/drive')

    Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

3.1.2 – Khám phá dữ liệu:

Tạo một đối tượng SparkSession để làm việc với dữ liệu bằng Apache Spark. Sử dụng nó để đọc dữ liệu từ một tệp CSV chứa dữ liệu về bệnh tiểu đường và sau đó in ra schema của DataFrame để hiển thị thông tin về cấu trúc dữ liệu.

```
[ ] #Tạo một SparkSession
    from pyspark.sql import SparkSession
    spark = SparkSession.builder.appName('ml-diabetes').getOrCreate()
    df = spark.read.csv('/content/drive/MyDrive/DoAn_BigData/Data/diabetes.csv', header = True, inferSchema = True)
    #In schema
    df.printSchema()

    root
     |-- Pregnancies: integer (nullable = true)
     |-- Glucose: integer (nullable = true)
     |-- BloodPressure: integer (nullable = true)
     |-- SkinThickness: integer (nullable = true)
     |-- Insulin: integer (nullable = true)
     |-- BMI: double (nullable = true)
     |-- DiabetesPedigreeFunction: double (nullable = true)
     |-- Age: integer (nullable = true)
     |-- Outcome: integer (nullable = true)
```

In một DataFrame của Pandas từ 5 dòng đầu tiên của DataFrame đã được đọc từ tệp CSV. Sau đó, nó chuyển vị (transpose) DataFrame đó để hiển thị dữ liệu dưới dạng cột thay vì dòng để có cái nhìn trực quan hơn về dữ liệu.

```
[ ] df.toPandas()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

Nhóm dữ liệu trong DataFrame dựa trên cột 'Outcome' - (Có-1/Không-0) để kiểm tra xem các lớp có cân bằng hoàn hảo không để giúp chúng ta nhận biết xem liệu có cần thực hiện các biện pháp cân bằng lớp hay không để đảm bảo rằng mô hình dự đoán của chúng ta hoạt động tốt trên cả hai lớp. Vì cân bằng lớp là một yếu tố quan trọng trong việc xây dựng mô hình dự đoán, vì nếu một trong hai lớp có quá ít mẫu so với lớp còn lại, mô hình có thể bị chệch hướng về lớp có số lượng mẫu lớn hơn.

```
[ ] #nhóm dữ liệu trong DataFrame dựa trên cột 'Outcome' -  (Có-1/Không-0)
    df.groupby('Outcome').count().toPandas()
```

	Outcome	count
0	1	268
1	0	500

Kết quả hiển thị cho thấy số lượng mẫu trong mỗi lớp của cột 'Outcome' trong dữ liệu tiểu đường. Cụ thể:

Lớp 1 (Có tiểu đường): Có 268 mẫu.

Lớp 0 (Không có tiểu đường): Có 500 mẫu.

Dữ liệu không cân bằng hoàn hảo vì số lượng mẫu trong hai lớp này không tương đương. Lớp 0 (không có tiểu đường) có số lượng mẫu lớn hơn nhiều so với lớp 1 (có tiểu đường). Điều này có thể ảnh hưởng đến quá trình xây dựng và đánh giá mô hình, vì mô hình có thể có xu hướng dự đoán chính xác hơn cho lớp có số lượng mẫu nhiều hơn, trong khi lớp khác có thể bị dự đoán sai lệch.

Sau đó chúng ta lập bảng số liệu thống kê chứa các thông số mô tả cho các thuộc tính số trong dữ liệu, bao gồm giá trị trung bình, độ lệch chuẩn, giá trị tối thiểu, giá trị tối đa và số lượng mẫu. Điều này giúp chúng ta có cái nhìn tổng quan về phân phối của các thuộc tính số trong dữ liệu tiểu đường.

```
[ ]  #thống kê của các đặc trưng dạng số trong DataFrame
     numeric_features = [t[0] for t in df.dtypes if t[1] == 'int']
     df.select(numeric_features).describe().toPandas().transpose()
```

	0	1	2	3	4
summary	count	mean	stddev	min	max
Pregnancies	768	3.8450520833333335	3.36957806269887	0	17
Glucose	768	120.89453125	31.97261819513622	0	199
BloodPressure	768	69.10546875	19.355807170644777	0	122
SkinThickness	768	20.536458333333332	15.952217567727642	0	99
Insulin	768	79.79947916666667	115.24400235133803	0	846
Age	768	33.240885416666664	11.760231540678689	21	81
Outcome	768	0.3489583333333333	0.476951377242799	0	1

Tạo ma trận scatter plot (biểu đồ phân tán) để thể hiện mối tương quan giữa các biến độc lập trong DataFrame dữ liệu về tiểu đường. Ma trận scatter plot là một biểu đồ sẽ hiển thị các scatter plot của từng cặp biến độc lập. Giúp chúng ta có cái nhìn tổng quan về các mối tương quan và tương quan giữa các cặp biến trong dữ liệu.

3.2 – CHUẨN BỊ DỮ LIỆU VÀ TRÍCH XUẤT ĐẶC TRƯNG:

3.2.1 – Xử lý dữ liệu bị thiếu:

Chuẩn bị dữ liệu và kỹ thuật trích xuất đặc trưng

```
[ ] from pyspark.sql.functions import isnull, when, count, col
    #Xử lý dữ liệu bị thiếu
    df.select([count(when(isnull(c), c)).alias(c) for c in df.columns]).show()
```

```
+-----------+-------+-------------+-------------+-------+---+------------------------+---+-------+
|Pregnancies|Glucose|BloodPressure|SkinThickness|Insulin|BMI|DiabetesPedigreeFunction|Age|Outcome|
+-----------+-------+-------------+-------------+-------+---+------------------------+---+-------+
|          0|      0|            0|            0|      0|  0|                       0|  0|      0|
+-----------+-------+-------------+-------------+-------+---+------------------------+---+-------+
```

Ở đây, không có giá trị nào bị thiếu trong các cột của DataFrame (số 0 trong bảng). Điều này cho thấy dữ liệu không có giá trị thiếu, hoặc ít nhất là không có giá trị thiếu nào được ghi đại diện bằng 'null'.

3.2.2 – Loại bỏ cột không cần thiết:

```
[ ] #Giảm cột không cần thiết
    df = df.drop('SkinThickness')
    df = df.drop('Insulin')
    df = df.drop('DiabetesPedigreeFunction')
    df = df.drop('Pregnancies')

    df.show()
```

```
+-------+-------------+----+---+-------+
|Glucose|BloodPressure| BMI|Age|Outcome|
+-------+-------------+----+---+-------+
|    148|           72|33.6| 50|      1|
|     85|           66|26.6| 31|      0|
|    183|           64|23.3| 32|      1|
|     89|           66|28.1| 21|      0|
|    137|           40|43.1| 33|      1|
|    116|           74|25.6| 30|      0|
|     78|           50|31.0| 26|      1|
|    115|            0|35.3| 29|      0|
|    197|           70|30.5| 53|      1|
|    125|           96| 0.0| 54|      1|
|    110|           92|37.6| 30|      0|
|    168|           74|38.0| 34|      1|
|    139|           80|27.1| 57|      0|
|    189|           60|30.1| 59|      1|
|    166|           72|25.8| 51|      1|
|    100|            0|30.0| 32|      1|
|    118|           84|45.8| 31|      1|
|    107|           74|29.6| 31|      1|
|    103|           30|43.3| 33|      0|
|    115|           70|34.6| 32|      1|
+-------+-------------+----+---+-------+
only showing top 20 rows
```

Loại bỏ đi cột 'SkinThickness', 'Insulin', 'DiabetesPedigreeFunction', 'Pregnancies'.

3.2.3 – Chuyển đổi các đặc trưng thành vecter:

Chuyển đổi các đặc trưng thành vector đặc trưng duy nhất bằng "VectorAssembler" giúp chúng ta chuẩn bị dữ liệu để có thể tiếp tục với các bước tiếp theo của quy trình học máy như huấn luyện mô hình, đánh giá mô hình và dự đoán trên dữ liệu mới.

```
# Chuyển đổi đặc trưng thành vector
required_features = ['Glucose',
                     'BloodPressure',
                     'BMI',
                     'Age'
                    ]

from pyspark.ml.feature import VectorAssembler

assembler = VectorAssembler(inputCols=required_features, outputCol='features')

transformed_data = assembler.transform(df)
transformed_data.show()
```

```
+-------+-------------+----+---+-------+--------------------+
|Glucose|BloodPressure| BMI|Age|Outcome|            features|
+-------+-------------+----+---+-------+--------------------+
|    148|           72|33.6| 50|      1|[148.0,72.0,33.6,...|
|     85|           66|26.6| 31|      0|[85.0,66.0,26.6,3...|
|    183|           64|23.3| 32|      1|[183.0,64.0,23.3,...|
|     89|           66|28.1| 21|      0|[89.0,66.0,28.1,2...|
|    137|           40|43.1| 33|      1|[137.0,40.0,43.1,...|
|    116|           74|25.6| 30|      0|[116.0,74.0,25.6,...|
|     78|           50|31.0| 26|      1|[78.0,50.0,31.0,2...|
|    115|            0|35.3| 29|      0|[115.0,0.0,35.3,2...|
|    197|           70|30.5| 53|      1|[197.0,70.0,30.5,...|
|    125|           96| 0.0| 54|      1|[125.0,96.0,0.0,5...|
|    110|           92|37.6| 30|      0|[110.0,92.0,37.6,...|
|    168|           74|38.0| 34|      1|[168.0,74.0,38.0,...|
|    139|           80|27.1| 57|      0|[139.0,80.0,27.1,...|
|    189|           60|30.1| 59|      1|[189.0,60.0,30.1,...|
|    166|           72|25.8| 51|      1|[166.0,72.0,25.8,...|
|    100|            0|30.0| 32|      1|[100.0,0.0,30.0,3...|
|    118|           84|45.8| 31|      1|[118.0,84.0,45.8,...|
|    107|           74|29.6| 31|      1|[107.0,74.0,29.6,...|
|    103|           30|43.3| 33|      0|[103.0,30.0,43.3,...|
|    115|           70|34.6| 32|      1|[115.0,70.0,34.6,...|
+-------+-------------+----+---+-------+--------------------+
only showing top 20 rows
```

3.3 – Xây dựng mô hình dự đoán, tạo model và huấn luyện:

3.3.1 – Phân tách dữ liệu thành train và test:

Bây giờ chúng ta hãy bắt đầu đào tạo mô hình. Trước tiên, chúng ta sẽ cần tách dữ liệu của mình bằng phương thức 'randomSplit'. 'transformed_data' thành hai phần: training_data (80%) và test_data (20%).

Sử dụng tham số ‘seed’ để đảm bảo rằng việc phân chia dữ liệu này có thể tái sản xuất được (nghĩa là cùng một seed sẽ dẫn đến cùng một phân chia dữ liệu).

```
[ ]  # Tách dữ liệu
     (training_data, test_data) = transformed_data.randomSplit([0.8,0.2], seed =2020)
     print("Số lượng dữ liệu tập huấn luyện: " + str(training_data.count()))
     print("Số lượng dữ liệu tập kiểm tra: " + str(test_data.count()))

     Số lượng dữ liệu tập huấn luyện: 620
     Số lượng dữ liệu tập kiểm tra: 148
```

Mục tiêu là xác định số lượng dữ liệu trong mỗi tập để đảm bảo rằng quá trình phân chia đã được thực hiện đúng cách và tỷ lệ phân chia đã được duy trì.

3.3.2 – Xây dựng, đánh giá độ chính xác của mô hình máy học:

Ở phần này chúng ta sẽ xây dựng 4 mô hình máy học khác nhau và so sánh để chọn ra mô hình tốt nhất cho bài toán dự đoán tiểu đường bằng PySpark.

a. Mô hình phân loại Rừng ngẫu nhiên:

Ở đây, mô hình ‘RandomForestClassifier’ được khởi tạo. Các tham số quan trọng bao gồm:

- ‘labelCol’: Xác định cột chứa nhãn (lớp) của dữ liệu, trong trường hợp này là cột 'Outcome'.
- ‘featuresCol’: Xác định cột chứa vector đặc trưng đã được tạo trước đó.
- ‘maxDepth’: Đây là một tham số kiểm soát độ sâu tối đa của các cây quyết định trong rừng ngẫu nhiên.

Mô hình Phân loại Rừng Ngẫu nhiên

```
[ ]  from pyspark.ml.classification import RandomForestClassifier

     rf = RandomForestClassifier(labelCol='Outcome',
                                 featuresCol='features',
                                 maxDepth=5)
     model = rf.fit(training_data)
     rf_predictions = model.transform(test_data)
```

Huấn luyện mô hình bằng phương thức fit trên tập dữ liệu huấn luyện (training_data) và sử dụng mô hình đã huấn luyện để dự đoán nhãn cho tập dữ liệu kiểm tra (test_data). Kết quả là một DataFrame mới có thêm cột 'prediction', chứa các dự đoán của mô hình. Mô hình được đánh giá dựa trên sự so sánh giữa dự đoán và nhãn thực tế trong tập kiểm tra.

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
#Đánh giá mô hình Phân loại Rừng Ngẫu nhiên
multi_evaluator = MulticlassClassificationEvaluator(labelCol = 'Outcome', metricName = 'accuracy')
print('Độ chính xác của mô hình Phân loại Rừng Ngẫu nhiên:', multi_evaluator.evaluate(rf_predictions))
```

```
Độ chính xác của mô hình Phân loại Rừng Ngẫu nhiên: 0.7567567567567568
```

Sử dụng ‘MulticlassClassificationEvaluator’ để đánh giá độ chính xác của mô hình phân loại Rừng Ngẫu nhiên trên dữ liệu kiểm tra. Trong đó:

- labelCol: Đây là cột chứa nhãn thực tế của dữ liệu (cột 'Outcome' trong trường hợp này).
- metricName: Đây là chỉ số đánh giá được sử dụng để đo lường hiệu suất của mô hình. Trong trường hợp này, sử dụng 'accuracy' (độ chính xác) làm chỉ số.
- .evaluate(rf_predictions): Phương thức này đánh giá mô hình dựa trên dữ liệu rf_predictions (có chứa dự đoán của mô hình).

Độ chính xác của mô hình được tính là khoảng 75.68%.

b. Mô hình phân loại Cây quyết định:

Ở mô hình này, chúng ta sử dụng ‘DecisionTreeClassifier’ để huấn luyện một mô hình cây quyết định trên dữ liệu huấn luyện và sau đó sử dụng mô hình đã huấn luyện để thực hiện dự đoán trên dữ liệu kiểm tra.

Mô hình phân loại Cây quyết định

```
from pyspark.ml.classification import DecisionTreeClassifier

dt = DecisionTreeClassifier(featuresCol = 'features', labelCol = 'Outcome', maxDepth = 3)
dtModel = dt.fit(training_data)
dt_predictions = dtModel.transform(test_data)
dt_predictions.select('Glucose', 'BloodPressure', 'BMI', 'Age', 'Outcome').show(10)
```

```
+-------+-------------+----+---+-------+
|Glucose|BloodPressure| BMI|Age|Outcome|
+-------+-------------+----+---+-------+
|     57|           80|32.8| 41|      0|
|     67|           76|45.3| 46|      0|
|     71|           48|20.4| 22|      0|
|     71|           78|33.2| 21|      0|
|     72|           78|31.6| 38|      0|
|     76|           60|32.8| 41|      0|
|     78|           50|31.0| 26|      1|
|     78|           88|36.9| 21|      0|
|     84|           64|35.8| 21|      0|
|     84|           82|38.2| 23|      0|
+-------+-------------+----+---+-------+
only showing top 10 rows
```

Một số điểm cần lưu ý:

- DecisionTreeClassifier: Đây là mô hình phân loại dựa trên cây quyết định. Các tham số như featuresCol, labelCol và maxDepth được sử dụng để cấu hình mô hình.
- featuresCol: Đây là tên của cột chứa các đặc trưng mà mô hình sẽ sử dụng để thực hiện dự đoán (trong trường hợp này là 'features').
- labelCol: Đây là tên của cột chứa nhãn của dữ liệu (trong trường hợp này là 'Outcome').
- maxDepth: Đây là độ sâu tối đa của cây quyết định.
- fit(training_data): Phương thức này được sử dụng để huấn luyện mô hình trên dữ liệu huấn luyện (training_data).
- transform(test_data): Phương thức này thực hiện dự đoán trên dữ liệu kiểm tra (test_data) bằng cách sử dụng mô hình đã huấn luyện và tạo ra cột 'prediction' trong kết quả.

```
[17] from pyspark.ml.evaluation import MulticlassClassificationEvaluator
     #Đánh giá mô hình phân loại Cây quyết định
     multi_evaluator = MulticlassClassificationEvaluator(labelCol = 'Outcome', metricName = 'accuracy')
     print('Độ chính xác của mô hình phân loại Cây quyết định:', multi_evaluator.evaluate(dt_predictions))

     Độ chính xác của mô hình phân loại Cây quyết định: 0.7702702702702703
```

Độ chính xác của mô hình được tính là khoảng 77%.

c. Mô hình Hồi quy Logistics:

Mô hình Hồi quy Logistics sử dụng 'LogisticRegression' để huấn luyện một mô hình hồi quy logistic trên dữ liệu huấn luyện, sau đó sử dụng mô hình đã huấn luyện để thực hiện dự đoán trên dữ liệu kiểm tra.

Mô hình hồi quy Logistic

```
[18] from pyspark.ml.classification import LogisticRegression

     lr = LogisticRegression(featuresCol = 'features', labelCol = 'Outcome', maxIter=10)
     lrModel = lr.fit(training_data)
     lr_predictions = lrModel.transform(test_data)
```

Dưới đây là một số chi tiết quan trọng:

- LogisticRegression: Đây là mô hình hồi quy logistic được sử dụng cho bài toán phân loại. Các tham số như featuresCol, labelCol và maxIter được sử dụng để cấu hình mô hình.
- featuresCol: Đây là tên của cột chứa các đặc trưng mà mô hình sẽ sử dụng để thực hiện dự đoán (trong trường hợp này là 'features').
- labelCol: Đây là tên của cột chứa nhãn của dữ liệu (trong trường hợp này là 'Outcome').
- maxIter: Số lần lặp tối đa trong quá trình tối ưu hóa.
- fit(training_data): Phương thức này được sử dụng để huấn luyện mô hình trên dữ liệu huấn luyện (training_data).

- transform(test_data): Phương thức này thực hiện dự đoán trên dữ liệu kiểm tra (test_data) bằng cách sử dụng mô hình đã huấn luyện và tạo ra cột 'prediction' trong kết quả.

```
[19] from pyspark.ml.evaluation import MulticlassClassificationEvaluator
     #Đánh giá mô hình hồi quy Logistic
     multi_evaluator = MulticlassClassificationEvaluator(labelCol = 'Outcome', metricName = 'accuracy')
     print('Độ chính xác của mô hình hồi quy Logistic:', multi_evaluator.evaluate(lr_predictions))

     Độ chính xác của mô hình hồi quy Logistic: 0.777027027027027
```

Độ chính xác của mô hình Hồi quy Logistics là khoảng 77.8%.

d. Mô hình phân loại Cây tang cường dốc:

Chúng ta sử dụng 'GBTClassifier' để huấn luyện một mô hình Gradient Boosting trên dữ liệu huấn luyện và sau đó sử dụng mô hình đã huấn luyện để thực hiện dự đoán trên dữ liệu kiểm tra.

```
from pyspark.ml.classification import GBTClassifier

gb = GBTClassifier(labelCol = 'Outcome', featuresCol = 'features')
gbModel = gb.fit(training_data)
gb_predictions = gbModel.transform(test_data)
```

Dưới đây là một số chi tiết liên quan:

- GBTClassifier: Đây là mô hình Gradient Boosting được sử dụng cho bài toán phân loại. Các tham số như labelCol và featuresCol được sử dụng để cấu hình mô hình.
- labelCol: Đây là tên của cột chứa nhãn của dữ liệu (trong trường hợp này là 'Outcome').
- featuresCol: Đây là tên của cột chứa các đặc trưng mà mô hình sẽ sử dụng để thực hiện dự đoán (trong trường hợp này là 'features').
- fit(training_data): Phương thức này được sử dụng để huấn luyện mô hình trên dữ liệu huấn luyện (training_data).

- transform(test_data): Phương thức này thực hiện dự đoán trên dữ liệu kiểm tra (test_data) bằng cách sử dụng mô hình đã huấn luyện và tạo ra cột 'prediction' trong kết quả.

Sau khi mô hình đã được huấn luyện và dự đoán, dữ liệu kết quả dự đoán (gb_predictions) có thể được sử dụng để thực hiện đánh giá hiệu suất của mô hình trên dữ liệu kiểm tra.

```
[21] from pyspark.ml.evaluation import MulticlassClassificationEvaluator
     #Đánh giá mô hình Phân loại Cây tăng cường dốc
     multi_evaluator = MulticlassClassificationEvaluator(labelCol = 'Outcome', metricName = 'accuracy')
     print('Độ chính xác của mô hình phân loại Cây tăng cường dốc:', multi_evaluator.evaluate(gb_predictions))

     Độ chính xác của mô hình phân loại Cây tăng cường dốc: 0.7567567567567568
```

Từ hình ảnh trên cho thấy độ chính xác của mô hình là khoảng 75.8%

3.3.3 – Đánh giá hiệu suất của mô hình:

Sau khi xây dựng và đánh giá các mô hình xong, chúng ta chọn ra mô hình Hồi quy Logicstics vì nó có độ chính xác cao nhất (khoảng 77.8%) để đánh giá hiệu suất.

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.mllib.evaluation import MulticlassMetrics

multi_evaluator = MulticlassClassificationEvaluator(labelCol='Outcome', metricName='accuracy')
accuracy = multi_evaluator.evaluate(lr_predictions)

predictionAndLabels = lr_predictions.select("prediction", "Outcome").rdd.map(lambda row: (float(row.prediction), float(row.Outcome)))

metrics = MulticlassMetrics(predictionAndLabels)
confusion_matrix = metrics.confusionMatrix().toArray()

precision = metrics.precision(1.0)  # Độ chính xác cho lớp tích cực
recall = metrics.recall(1.0)        # Độ nhạy cho lớp tích cực
f1_score = metrics.fMeasure(1.0)    # Điểm F1 cho lớp tích cực

print("Accuracy:", accuracy)
print("Confusion Matrix:\n", confusion_matrix)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1_score)
```

Để đánh giá hiệu suất của mô hình chúng ta cần cái nhìn tổng quan về:

- Confusion Matrix (Ma trận nhầm lẫn
- Precision (Độ chính xác
- Recall (Độ nhạy
- F1-score

3.4 – KẾT LUẬN:

3.2.1 – KẾT QUẢ ĐẠT ĐƯỢC:

Từ các kết quả của 4 mô hình cho ta thấy rõ mô hình Hồi quy Logistics đạt mức chính xác cao nhất. Khoảng 77.8% so với 3 mô hình còn lại.

Qua kết quả xây dựng mô hình, tạo model, dự đoán và đánh giá mô hình. Chúng ta được kết quả như sau:

- **Accuracy:** 0.777027027027027
- **Confusion Matrix:** [[84. 10.] [23. 31.]]
- **Precision:** 0.7560975609756098
- **Recall:** 0.5740740740740741
- **F1-score:** 0.6526315789473683

Dựa vào những kết quả ở trên, chúng ta có thể đưa ra kết luận như sau:

1. **Accuracy:** Độ chính xác tổng thể của mô hình là khoảng 77.70%, có nghĩa là mô hình dự đoán đúng khoảng 77.70% trên tất cả các trường hợp kiểm tra.
2. **Confusion Matrix:**
    - True Positive (TP): Có 31 trường hợp được dự đoán là positive và thực tế cũng là positive.
    - True Negative (TN): Có 84 trường hợp được dự đoán là negative và thực tế cũng là negative.
    - False Positive (FP): Có 10 trường hợp được dự đoán là positive nhưng thực tế lại là negative.
    - False Negative (FN): Có 23 trường hợp được dự đoán là negative nhưng thực tế lại là positive.
3. **Precision:** Độ chính xác cho lớp positive (có bệnh) là khoảng **75.61%**. Trong các trường hợp mà mô hình dự đoán là có bệnh, khoảng **75.61%** thực sự là có bệnh.

4. **Recall:** Độ nhạy cho lớp positive là khoảng **57.41%**. Mô hình bắt được khoảng 57.41% tổng số trường hợp có thực tế là có bệnh.
5. **F1-score:** Điểm F1 cho lớp positive là khoảng **65.26%**. Điểm F1 là sự kết hợp trung bình điều hòa giữa Precision và Recall, giúp đánh giá cân bằng giữa việc dự đoán chính xác các trường hợp positive và việc bắt được nhiều trường hợp positive.

Tổng kết, mô hình có độ chính xác tổng thể khá tốt, nhưng có thể cần cải thiện độ nhạy (Recall) để bắt được nhiều trường hợp có bệnh hơn. Điều này có thể thực hiện bằng cách điều chỉnh ngưỡng quyết định hoặc cân nhắc sử dụng các phương pháp cân bằng lớp dữ liệu (như tăng số lượng dữ liệu cho lớp thiểu số) để cải thiện khả năng dự đoán lớp positive.

3.2.2 – HẠN CHẾ VÀ HƯỚNG PHÁT TRIỂN:

a. Hạn chế:

Dữ liệu không cân bằng: Trong trường hợp dữ liệu không cân bằng, tỷ lệ giữa các lớp không đồng đều. Điều này có thể làm cho mô hình có hiệu suất tốt cho lớp đa số nhưng kém cho lớp thiểu số. Việc xử lý dữ liệu không cân bằng là một thách thức và có thể ảnh hưởng đến khả năng dự đoán chính xác.

Phụ thuộc vào đặc trưng đo lường: Mô hình phụ thuộc vào các đặc trưng đo lường như Glucose, BloodPressure, BMI và Age. Nếu không có các đặc trưng đo lường đầy đủ và chính xác, mô hình có thể không hoạt động tốt.

Không xem xét các yếu tố khác: Dự án tập trung vào dự đoán dựa trên các đặc trưng y tế cá nhân. Tuy nhiên, có nhiều yếu tố khác cũng có thể ảnh hưởng đến bệnh tiểu đường như di truyền, lối sống, chế độ ăn uống, v.v. Không xem xét các yếu tố này có thể là hạn chế.

Giới hạn của các mô hình: Mặc dù đã thử nghiệm nhiều loại mô hình như Random Forest, Decision Tree, Logistic Regression và Gradient Boosting, nhưng vẫn có thể có các mô hình khác hoặc kỹ thuật mới có thể cải thiện hiệu suất dự đoán.

Khả năng ứng dụng thực tế: Hiệu suất của mô hình trong môi trường thực tế có thể khác so với trên tập kiểm tra, do điều kiện thực tế phức tạp hơn và có nhiều yếu tố không xác định.

Yếu tố người dùng: Dự đoán bệnh tiểu đường và quản lý bệnh đòi hỏi sự hợp tác và tuân thủ của bệnh nhân. Mô hình chỉ có thể đưa ra dự đoán, nhưng không thể kiểm soát hành vi và quyết định của người dùng.

b. Hướng phát triển:

Thu thập dữ liệu bổ sung: Thu thập thêm dữ liệu từ nhiều nguồn khác nhau để bổ sung cho tập dữ liệu hiện có. Điều này có thể cung cấp thông tin bổ sung về các yếu tố di truyền, lối sống, chế độ ăn uống, tình trạng sức khỏe tổng thể và các biến đổi theo thời gian.

Xử lý dữ liệu không cân bằng: Để cải thiện hiệu suất của mô hình trên lớp thiểu số, chúng ta có thể sử dụng kỹ thuật xử lý dữ liệu không cân bằng như oversampling, undersampling hoặc sử dụng các thuật toán dự đoán dựa trên trọng số.

Tăng cường tính tổng quát của mô hình: Thử nghiệm và so sánh với nhiều mô hình hơn, bao gồm các mô hình học sâu như mạng nơ-ron nhân tạo. Điều này có thể giúp cải thiện khả năng dự đoán và đảm bảo tính tổng quát của mô hình.

Tinh chỉnh tham số mô hình: Thử nghiệm các giá trị tham số khác nhau cho các mô hình đã sử dụng để tìm ra các cấu hình tốt nhất. Việc này có thể cải thiện hiệu suất dự đoán và độ chính xác của mô hình.

Kết hợp nhiều mô hình: Kết hợp dự đoán từ nhiều mô hình khác nhau có thể giúp cải thiện độ chính xác và ổn định của dự đoán. Các phương pháp như bình chọn đa số, bình chọn mềm hoặc stacking có thể được thử nghiệm.

Phân tích thêm về yếu tố ảnh hưởng: Nghiên cứu sâu hơn về tác động của các đặc trưng riêng lẻ đối với việc dự đoán bệnh tiểu đường có thể giúp hiểu rõ hơn về cơ chế và tác nhân gây ra bệnh.

Xem xét mô hình giải thích: Sử dụng các phương pháp giải thích mô hình như SHAP, LIME để hiểu rõ hơn về cách mô hình ra quyết định và tác động của từng đặc trưng đối với dự đoán.

Phát triển ứng dụng thực tế: Xây dựng một ứng dụng hoặc giao diện người dùng cho việc dự đoán bệnh tiểu đường có thể giúp người dùng theo dõi và quản lý tình trạng sức khỏe của họ.

Kiểm tra trong môi trường thực tế: Kiểm tra hiệu suất và khả năng áp dụng thực tế của mô hình trong môi trường y tế thực tế và thu thập phản hồi từ bệnh nhân và chuyên gia y tế.

Tóm lại, dự án này có thể tiếp tục phát triển để cải thiện độ chính xác và ứng dụng thực tế trong việc dự đoán và quản lý bệnh tiểu đường.

TÀI LIỆU THAM KHẢO

1. diabetes.csv | Kaggle
2. PySpark Tutorial For Beginners | Python Examples - Spark By {Examples} (sparkbyexamples.com)
3. Thầy Hồ Khôi, slide bài giảng môn Dữ liệu lớn, khoa CNTT, trường đại học Nguyễn Tất Thành.

LINK SOURCE CODE:

https://drive.google.com/drive/folders/1sNx3VM2l6n3d4PzyGohB2t5gvo1JMEL-?usp=sharing