BỘ GIÁO DỤC VÀ ĐÀO TẠO

ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN HÀ NỘI

BÁO CÁO MÔN HỌC

QUẢN TRỊ DỮ LIỆU LỚN

ĐỀ TÀI: REAL-TIME FRAUD FINANCE DETECTION IN MEDICAL

NHÓM 18

Thành viên:	**Phạm Hùng**	**21000143**
	Nguyễn Tuấn Nguyên	**21000412**
	Nguyễn Trí Mạnh	**20002069**

Giảng viên: **Phạm Tiến Lâm**

Đặng Văn Báu

INTRODUCTION

Chăm sóc sức khỏe là một ngành quan trọng ở Hoa Kỳ với các chương trình do cả tư nhân và chính phủ điều hành. Chi phí của các dịch vụ y tế tiếp tục tăng lên, ở một mức độ nào đó do dân số người già ngày càng tăng. Chi tiêu bảo hiểm y tế của Hoa Kỳ từ năm 2012 đến năm 2014 đã tăng 6,7% để đạt 3 nghìn tỷ USD và chi tiêu của Medicare chiếm 20% tổng chi tiêu cho dịch vụ con người ở Hoa Kỳ với khoảng 600 tỷ USD. Nhóm dân số già đang gia tăng này, cùng với chi phí ngày càng tăng của Medicare, cần các thỏa thuận cắt giảm chi phí, trong đó việc giảm

trình bày sai là một cách giúp phục hồi chi phí và giảm dần các khoản thanh toán lớn. Tác động của việc tống tiền chăm sóc sức khỏe được đánh giá là từ 3% đến 10% chi tiêu tuyệt đối cho dịch vụ con người của đất nước và tiếp tục ảnh hưởng tiêu cực đến chương trình Medicare và những người nhận chương trình này (NHCAA 2017). Chính phủ đã giới thiệu các dự án như Lực lượng tấn công gian lận Medicare (OIG 2017), được ủy quyền để giúp chống lại hành vi xuyên tạc, tuy nhiên, những nỗ lực được tiến hành dự kiến sẽ có nhiều khả năng làm giảm tác động của việc tống tiền. Đây là phần trình bày và nền tảng thiết yếu cho nhiệm vụ của chúng ta; tương tự như vậy, nó cho thấy rằng có rất nhiều cơ hội kinh doanh mở ra cho các hệ thống Phát hiện Gian lận của Medicare. Dự án này cam kết xây dựng các giải pháp dữ liệu lớn với các ứng dụng quan trọng tại điểm giao nhau của ngành bảo vệ và chăm sóc sức khỏe. Dự án này sẽ xây dựng mô hình Phát hiện gian lận Medicare để điều tra thông tin mở và dự đoán/xác định quyền lợi của nhà cung cấp Medicare giả dựa trên các thiết kế trình bày sai, kiểm tra kỳ lạ và đo lường phân khúc địa lý. Mục tiêu của chúng ta là tạo ra một mô hình dự đoán gian lận trong ngành bảo hiểm y tế bằng cách sử dụng phân tích bất thường và số liệu địa lý nhân khẩu học. Sẽ có một số lợi ích nếu chúng ta có thể phát hiện ra hành vi gian lận một cách chính xác. Dưới đây là những ưu điểm sau:

• Bảo vệ và khấu trừ chi phí Nhận biết trước hành vi tống tiền không còn truy đuổi những kẻ vi phạm pháp luật sau khi đã thực hiện trả góp, bên cạnh các chi phí hợp pháp và sâu sắc. Đó chỉ là một cách tiếp cận thông minh hơn nhiều để làm việc cùng nhau. Hơn nữa, bất kể quỹ dự trữ chi phí đó có ảnh hưởng đến mối quan tâm chính của doanh nghiệp bạn hay của một bộ phận mở mà bạn được nhận hay không, kết quả phải là những kết quả có thể chấp nhận được.

• Ngăn chặn gian lận trở nên dễ dàng Khi mô hình của chúng tôi nhận ra, chúng tôi có thể giúp cung cấp thông tin bổ sung để truy cập bất kỳ hành vi gian lận nào. Nó sẽ giúp ích trong việc tìm kiếm các mẫu có thể bằng cách so sánh lẫn nhau.

• Người trả tiền có thể duy trì việc tuân thủ các quy định thanh toán nhanh chóng của chính phủ Biện pháp "theo đuổi" thủ công các vụ việc, bất kể trước hay sau trả góp, về cơ bản đã giảm bớt. Việc kiểm tra có thể diễn ra dần dần khi các trường hợp được đưa ra và mục tiêu được hoàn thành nhanh hơn rất nhiều. Cho dù bạn là người quản lý đô la mở hay phúc lợi tài chính của doanh nghiệp quản lý của chính bạn, thì hệ thống lập trình kiểm tra dự đoán sẽ là tín hiệu tốt.

PROBLEM STATEMENT

Xây dựng mô hình machine learning giúp dự đoán gian lận trong ngành bảo hiểm y tế bằng cách sử dụng thuật toán phân loại và phân loại theo thời gian thực. Chính phủ có thể sử dụng công cụ này để mang lại lợi ích cho bệnh nhân, nhà thuốc, bác sĩ, cuối cùng giúp đạt được uy tín trong ngành, giải quyết chi phí chăm sóc sức khỏe ngày càng tăng và xử lý tác động của gian lận. Trình bày sai về dịch vụ y tế và những gian lận là một vấn đề cơ bản gây ra tổn thất tài chính lớn trong ngành bảo hiểm và Medicare/Medicaid. Trung tâm Dịch vụ Medicare và Medicaid (CMS-The Centres for Medicare and Medicaid Services) đã sắp xếp các chương trình Medicare Part D từ năm 2006. CMS dựa vào đó để xác định và ngăn chặn hành vi tống tiền, lãng phí và ngược đãi trong chương trình Part D. Tuy nhiên, bằng cách sử dụng các kỹ thuật thông thường, các điểm gian lận có thể không bị phát hiện hoặc việc điều tra gian lận thủ công gây ra nhiều tốn kém. Theo báo cáo của Văn phòng Tổng Thanh tra: Từ năm 2006, Gian lận Medicare đã nhanh chóng lan rộng. Các âm mưu tống tiền bao gồm bốn loại sau:

• Lừa đảo bởi Nhà cung cấp dịch vụ (Bác sĩ, bệnh viện, nhà thuốc)

• Lừa đảo bởi người đăng ký bảo hiểm (bệnh nhân hoặc người nhà của bệnh nhân)

• Lừa đảo bởi hãng bảo hiểm

• Âm mưu lừa đảo (có liên quan đến tất cả các bên)

Bộ dữ liệu mà chúng tôi sử dụng bao gồm: Part D Prescriber Dataset, Excluded (LEIE) dataset, Payment Received dataset Dataset Link.

Mục tiêu chính của dự án này là:

• Xây dựng Mô hình dữ liệu cơ bản để hiển thị các kết nối giữa các bộ dữ liệu đặc biệt và phân biệt các đặc điểm chính để nhận dạng gian lận.

• Xây dựng mô hình AI để nhận ra gian lận dựa trên các điểm nổi bật khác nhau: Nhà cung cấp dịch vụ (Bác sĩ) , Nhà thuốc), Người hỗ trợ bảo hiểm (bệnh nhân), Phân khúc địa lý và thường lạm dụng thuốc. Cụ thể chúng ta sẽ nhận diện xem nhà cung cấp dịch vụ chăm sóc sức khoẻ(NPI) nào đã có hành vi gian lận, dựa trên số liệu về những đặc tính khác nhau của NPI đó.

• Xây dựng hệ thống nhận diện gian lận theo thời gian thực dựa trên Kafka – một nền tảng xử lí streaming data.

METHODOLGY

A. DATABASE SELECTION

1. CMS Prescriber Data

Tên bộ dữ liệu: Medicare Part D Prescribers - by Provider 2021

Giới thiệu chung: Thông tin về thuốc theo toa được cung cấp cho những người thụ hưởng Medicare đã đăng ký trong Phần D (Bảo hiểm thuốc theo toa), bởi các bác sĩ và nhà cung cấp dịch vụ chăm sóc sức khỏe khác, được tổng hợp bởi nhà cung cấp.

Nguồn dữ liệu: Data source Centers for Medicare & Medicaid Services

Tần suất cập nhật dữ liệu: Hàng năm

Dữ liệu mới nhất có sẵn: 2021

Tập dữ liệu về Medicare Part D Prescribers - by Provider chứa thông tin về thuốc theo toa do từng bác sĩ và nhà cung cấp dịch vụ chăm sóc sức khỏe khác kê toa và được thanh toán theo Chương trình Thuốc Theo toa Medicare Phần D. Tập dữ liệu xác định các nhà cung cấp theo Mã nhận dạng nhà cung cấp quốc gia (NPI) của họ và tóm tắt cho mỗi người kê đơn tổng số đơn thuốc đã được cấp, bao gồm các đơn thuốc ban đầu và bất kỳ lần mua thêm nào cũng như tổng chi phí thuốc.

Bộ dữ liệu về Medicare Part D Prescribers - by Provider (Bảng tóm tắt về người kê đơn Phần D) bao gồm việc sử dụng thuốc tổng thể (yêu cầu bồi thường, số lượng thuốc được tiêu chuẩn hóa trong 30 ngày và lượng cung cấp trong ngày), chi phí thuốc và số lượng người thụ hưởng do NPI tổ chức. Việc sử dụng thuốc, chi phí thuốc và số lượng người thụ hưởng cũng được đưa vào từng phân loại sau:

- Người thụ hưởng từ 65 tuổi trở lên
- Thuốc chính hiệu, thuốc gốc và các loại thuốc khác;
- Thuốc theo toa Medicare Advantage (MAPD) và Chương trình thuốc theo toa độc lập (PDP)
- Trợ cấp cho người thu nhập thấp (LIS) và không trợ cấp cho người thu nhập thấp (nonLIS)

- Thuốc phiện, thuốc phiện tác dụng kéo dài, thuốc kháng sinh và thuốc chống loạn thần ở người cao tuổi

Ngoài ra, các đặc điểm nhân khẩu học và sức khỏe của người thụ hưởng cũng được cung cấp bao gồm tuổi tác, giới tính, chủng tộc, quyền lợi Medicare và Medicaid cũng như điểm rủi ro.

Lưu ý: Tập dữ liệu đầy đủ này chứa nhiều bản ghi hơn mức hầu hết các chương trình bảng tính có thể xử lý, điều này sẽ dẫn đến việc tải dữ liệu không đầy đủ. Việc sử dụng cơ sở dữ liệu hoặc phần mềm thống kê là bắt buộc.

```
In [553…  partD_pd.shape

Out[553…  (1287454, 85)
```

Bộ dữ liệu này có 85 cột và 1287454 dòng. Tên các cột được thể hiện trong hình dưới

```
In [554…  partD_pd.columns

Out[554…  Index(['PRSCRBR_NPI', 'Prscrbr_Last_Org_Name', 'Prscrbr_First_Name',
         'Prscrbr_MI', 'Prscrbr_Crdntls', 'Prscrbr_Gndr', 'Prscrbr_Ent_Cd',
         'Prscrbr_St1', 'Prscrbr_St2', 'Prscrbr_City', 'Prscrbr_State_Abrvtn',
         'Prscrbr_State_FIPS', 'Prscrbr_zip5', 'Prscrbr_RUCA',
         'Prscrbr_RUCA_Desc', 'Prscrbr_Cntry', 'Prscrbr_Type',
         'Prscrbr_Type_src', 'Tot_Clms', 'Tot_30day_Fills', 'Tot_Drug_Cst',
         'Tot_Day_Suply', 'Tot_Benes', 'GE65_Sprsn_Flag', 'GE65_Tot_Clms',
         'GE65_Tot_30day_Fills', 'GE65_Tot_Drug_Cst', 'GE65_Tot_Day_Suply',
         'GE65_Bene_Sprsn_Flag', 'GE65_Tot_Benes', 'Brnd_Sprsn_Flag',
         'Brnd_Tot_Clms', 'Brnd_Tot_Drug_Cst', 'Gnrc_Sprsn_Flag',
         'Gnrc_Tot_Clms', 'Gnrc_Tot_Drug_Cst', 'Othr_Sprsn_Flag',
         'Othr_Tot_Clms', 'Othr_Tot_Drug_Cst', 'MAPD_Sprsn_Flag',
         'MAPD_Tot_Clms', 'MAPD_Tot_Drug_Cst', 'PDP_Sprsn_Flag', 'PDP_Tot_Clms',
         'PDP_Tot_Drug_Cst', 'LIS_Sprsn_Flag', 'LIS_Tot_Clms', 'LIS_Drug_Cst',
         'NonLIS_Sprsn_Flag', 'NonLIS_Tot_Clms', 'NonLIS_Drug_Cst',
         'Opioid_Tot_Clms', 'Opioid_Tot_Drug_Cst', 'Opioid_Tot_Suply',
         'Opioid_Tot_Benes', 'Opioid_Prscrbr_Rate', 'Opioid_LA_Tot_Clms',
         'Opioid_LA_Tot_Drug_Cst', 'Opioid_LA_Tot_Suply', 'Opioid_LA_Tot_Benes',
         'Opioid_LA_Prscrbr_Rate', 'Antbtc_Tot_Clms', 'Antbtc_Tot_Drug_Cst',
         'Antbtc_Tot_Benes', 'Antpsyct_GE65_Sprsn_Flag',
         'Antpsyct_GE65_Tot_Clms', 'Antpsyct_GE65_Tot_Drug_Cst',
         'Antpsyct_GE65_Bene_Suprsn_Flag', 'Antpsyct_GE65_Tot_Benes',
         'Bene_Avg_Age', 'Bene_Age_LT_65_Cnt', 'Bene_Age_65_74_Cnt',
         'Bene_Age_75_84_Cnt', 'Bene_Age_GT_84_Cnt', 'Bene_Feml_Cnt',
         'Bene_Male_Cnt', 'Bene_Race_Wht_Cnt', 'Bene_Race_Black_Cnt',
         'Bene_Race_Api_Cnt', 'Bene_Race_Hspnc_Cnt', 'Bene_Race_Natind_Cnt',
         'Bene_Race_Othr_Cnt', 'Bene_Dual_Cnt', 'Bene_Ndual_Cnt',
         'Bene_Avg_Risk_Scre'],
        dtype='object')
```

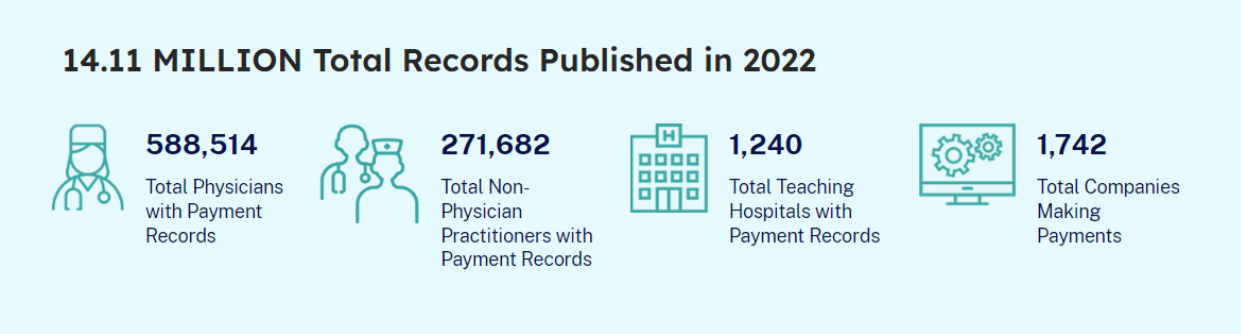
Xem trước bộ dữ liệu:

PRSCRBR_NPI	Prscrbr_Last_Org_Name	Prscrbr_First_Name	Prscrbr_MI	Prscrbr_Crdntls	Prscrbr_Gndr	Prscrbr_Ent_Cd	Pr
1003000126	Enkeshafi	Ardalan		M.D.	M	I	
1003000142	Khalil	Rashid		M.D.	M	I	
1003000167	Escobar	Julio	E	DDS	M	I	
1003000175	Reyes-Vasquez	Belinda		D.D.S.	F	I	
1003000423	Velotta	Jennifer	A	M.D.	F	I	
1003000480	Rothchild	Kevin	B	MD	M	I	
1003000522	Weigand	Frederick	J	MD	M	I	
1003000530	Semonche	Amanda	M	DO	F	I	

2. Payment Data 2022

Dữ liệu open payments là thông tin có thể truy cập công khai về các khoản thanh toán và chuyển giao giá trị mà tổ chức báo cáo thực hiện cho người nhận được bảo hiểm . Các đơn vị báo cáo có trách nhiệm theo dõi các giao dịch tài chính này hàng năm và phải gửi dữ liệu này đến Trung tâm Dịch vụ Medicare & Medicaid (CMS).

CMS xuất bản bộ dữ liệu đầy đủ cho năm chương trình trước đó hàng năm vào hoặc trước ngày 30 tháng 6 và làm mới dữ liệu vào tháng 1 hàng năm. Dữ liệu có thể được truy cập công khai thông qua Công cụ tìm kiếm open payments . Các bộ dữ liệu năm chương trình riêng lẻ có sẵn trên Trang tải xuống bộ dữ liệu.



Bộ dữ liệu có thông tin về thanh toán gồm 558514 bác sĩ, 271682 là người bán thuốc (không phải là bác sĩ nhưng bán thuốc theo đơn), 1240 bệnh viện và 1742 công ty.

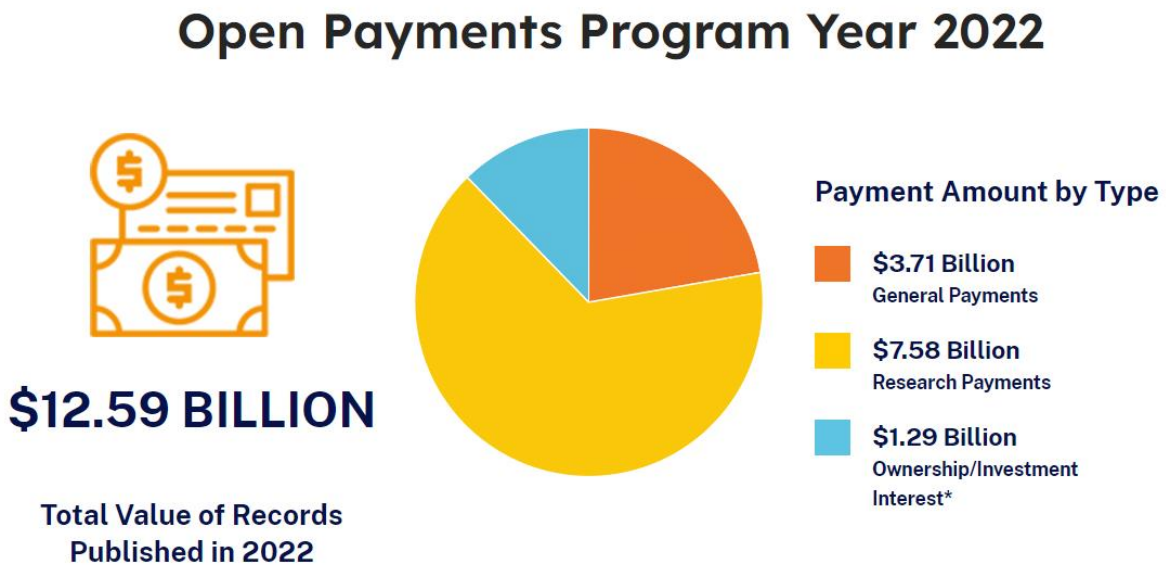
Dữ liệu được báo cáo có ba loại thanh toán:

- General Payments: Thanh toán hoặc chuyển giao giá trị không liên quan đến thỏa thuận nghiên cứu hoặc đề cương nghiên cứu

- Research Payments: Thanh toán hoặc chuyển giao giá trị được thực hiện liên quan đến thỏa thuận nghiên cứu chính thức hoặc đề cương nghiên cứu
- Ownership and Investment Interests: Thông tin về quyền sở hữu hoặc lợi ích đầu tư mà bác sĩ hoặc thành viên gia đình trực tiếp của họ có với đơn vị báo cáo

Các khoản thanh toán này có thể được báo cáo như sau:

- Direct Payments: Thanh toán trực tiếp cho người nhận được bảo hiểm
- Indirect Payments : Thanh toán gián tiếp cho người nhận được bảo hiểm thông qua một bên trung gian như hiệp hội chuyên khoa y tế
- Third Party Payments: Được chỉ định bởi người nhận được bảo hiểm để thanh toán cho bên khác



Tổng cộng tất cả thanh toán có giá trị là 12.59 tỉ đô, trong đó bao gồm 3.71 tỉ đô là các thanh toán chung, 7.58 tỉ đô là các thanh toán nghiên cứu và 1.29 tỉ đô là các thanh toán liên quan đến quyền sở hữu và lợi ích đầu tư. Thông tin về các loại thanh toán đã được giải thích ở bên trên.

3. List of Excluded Individuals/Entities (LEIE) 2023

The Office of Inspector General (OIG) là tổ chức chống lãng phí, gian lận và lạm dụng cũng như nâng cao hiệu quả của Medicare, Medicaid và hơn 100 Bộ Y tế & Dịch vụ Nhân sinh khác. OIG có quyền loại trừ các cá nhân và tổ chức khỏi các

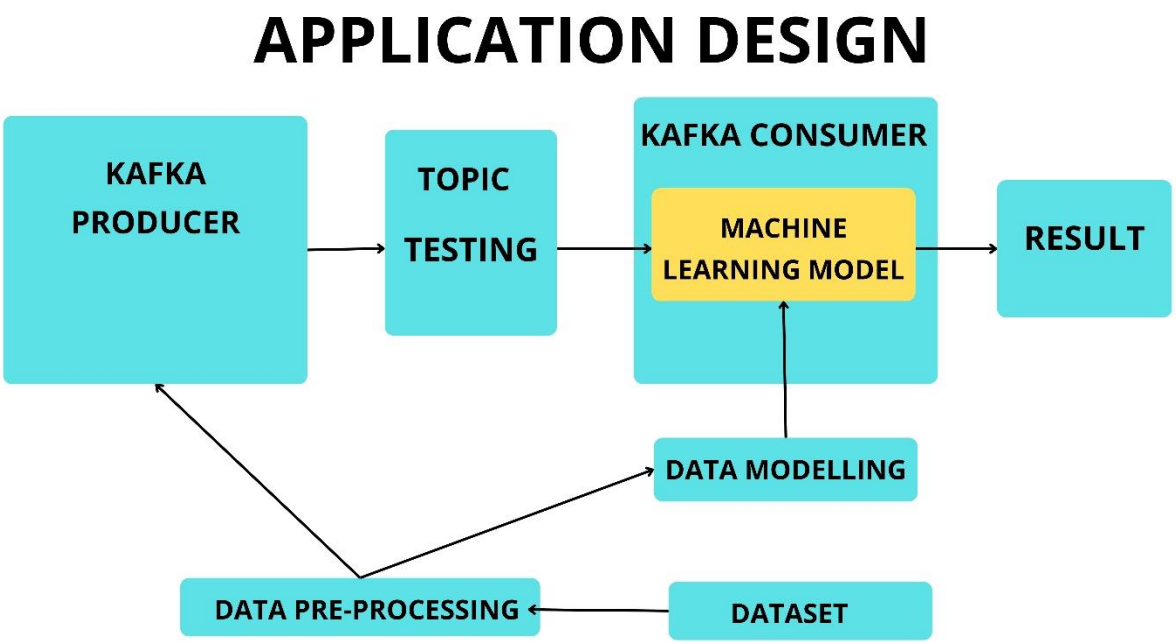
chương trình chăm sóc sức khỏe do Liên bang tài trợ vì nhiều lý do, bao gồm cả việc bị kết tội gian lận Medicare hoặc Medicaid. Những người bị loại trừ sẽ không nhận được khoản thanh toán nào từ các chương trình chăm sóc sức khỏe Liên bang cho bất kỳ hạng mục hoặc dịch vụ nào họ cung cấp, đặt hàng hoặc kê đơn. Điều này bao gồm những khoản cung cấp phúc lợi y tế được tài trợ trực tiếp hoặc gián tiếp bởi Hoa Kỳ (trừ Chương trình Phúc lợi Y tế Nhân viên Liên bang)

Bộ dữ liệu này được OIG cung cấp dữ liệu về danh sách các cá nhân, tổ chức vi phạm về tài chính trong lĩnh vực chăm sóc sức khỏe. Tên các trường dữ liệu được thể hiện trong hình dưới.

	FIELD VALUE
	LASTNAME
	FIRSTNAME
	MIDNAME
	BUSNAME
	GENERAL
	SPECIALTY
	UPIN
	NPI
	DOB
	ADDRESS
	CITY
	STATE
	ZIP CODE
	EXCLTYPE
	EXCLDATE
	REINDATE
	WAIVERDATE
	WAIVERSTATE

B. APPLICATION DESIGN

Về phần thiết kế ứng dụng, đầu tiên ta thực hiện nhập dataset sau đó thực hiện tiền xử lí dữ liệu. Dữ liệu sau khi xử lí sẽ được sử dụng để xây dựng mô hình machine learning. Lúc này, ta sẽ xây dựng mô hình machine và sử dụng chúng để nhận diện gian lận. Nhiệm vụ của kafka là nhận dữ liệu đã qua xử lí và đưa dữ liệu đó vào mô hình machine learning để dự đoán gian lận theo thời gian thực.



Phần tiền xử lí dữ liệu sẽ xử lí dữ liệu trống, loại bỏ các giá trị trùng lặp, gộp các dataset, chuyển đổi dữ liệu non-numerical sang numerical, chuẩn hóa dữ liệu, thực hiện cân bằng mẫu (class balancing).

Phần data modelling sẽ xây dựng mô hình dựa vào dữ liệu đã qua xử lí. Các mô hình machine learning được xây dựng bao gồm 3 mô hình: Logistic Regression, Random Forest Classifier, Extra Tree Classifier và Gradient Boosting Classifier.

Cuối cùng Kafka sẽ nhận dữ liệu đã qua xử lí và sử dụng mô hình đã xây dựng để trả ra kết quả dự đoán.

Tóm lại, ứng dụng này cung cấp khả năng nhận diện gian lận theo thời gian thực dựa vào nền tảng Kafka.

C. DATA PRE-PROCESSING

1. Impute missing data

```
full_dataset
```

	npi	specialty_description	total_drug_cost	total_claim	total_day_supply	total_beneficiaries	total_payment_sum	is_fraud
0	1003000126	Internal Medicine	81553.44	1123	74080	358.0	NaN	NaN
1	1003000142	Anesthesiology	37841.04	1493	43944	299.0	NaN	NaN
2	1003000167	Dentist	221.66	47	455	35.0	NaN	NaN
3	1003000175	Dentist	125.82	20	161	11.0	NaN	NaN
4	1003000423	Obstetrics & Gynecology	20757.65	206	10231	66.0	35.88	NaN
...	...	...	...	...	...	...	...	...
287450	1992999650	Dentist	498.08	74	1173	30.0	NaN	NaN
287451	1992999692	Pharmacist	2167.29	15	282	13.0	NaN	NaN
287452	1992999817	Orthopaedic Surgery	248.81	25	219	17.0	NaN	NaN
287453	1992999825	Otolaryngology	7712.09	206	4636	102.0	NaN	NaN
287454	1992999874	Internal Medicine	16239.45	304	6964	103.0	NaN	NaN

Như ta thấy trong dataset , có 2 trường là “total_payment_sum” và “is_fraud” xuất hiện những giá trị “NaN” . Đây là những giá trị còn bị thiếu vậy nên chúng ta phải quy chuẩn lại những giá trị này thành những con số cụ thể.

Filling NaN values with 0, in here the rows whose column "is_fraud" is NaN would be filled with the value 0.

```
full_dataset.fillna({"is_fraud": 0}, inplace = True)
```

```
full_dataset.head()
```

	npi	specialty_description	total_drug_cost	total_claim	total_day_supply	total_beneficiaries	total_payment_sum	is_fraud
0	1003000126	Internal Medicine	81553.44	1123	74080	358.0	NaN	0.0
1	1003000142	Anesthesiology	37841.04	1493	43944	299.0	NaN	0.0
2	1003000167	Dentist	221.66	47	455	35.0	NaN	0.0
3	1003000175	Dentist	125.82	20	161	11.0	NaN	0.0
4	1003000423	Obstetrics & Gynecology	20757.65	206	10231	66.0	35.88	0.0

Hình ảnh trên là dataset sau khi quy chuẩn các giá trị NaN ở cột "is_fraud" trở thành 0 bằng hàm fillna(). Tiếp theo chúng ta sẽ xử lý các giá trị NaN ở cột total_payment_sum.

The column "total_payment_sum" NaN values will be filled by the median value of column

```
full_dataset['total_payment_sum'] = full_dataset['total_payment_sum'].fillna(
    full_dataset["total_payment_sum"].median())
```

```
full_dataset.head()
```

	npi	specialty_description	total_drug_cost	total_claim	total_day_supply	total_beneficiaries	total_payment_sum	is_fraud
0	1003000126	Internal Medicine	81553.44	1123	74080	358.0	55.57	0.0
1	1003000142	Anesthesiology	37841.04	1493	43944	299.0	55.57	0.0
2	1003000167	Dentist	221.66	47	455	35.0	55.57	0.0
3	1003000175	Dentist	125.82	20	161	11.0	55.57	0.0
4	1003000423	Obstetrics & Gynecology	20757.65	206	10231	66.0	35.88	0.0

Các giá trị NaN ở cột total_payment_sum đã được thay thế bằng các giá trị trung vị (median) tương ứng .

Như vậy chúng ta đã hoàn thành bước quy chuẩn lại những giá trị khuyết. Tiếp theo chúng ta sẽ đến với bước xóa bỏ các giá trị trùng lặp .

2. Removing duplicates

```
partD_pd0.head()
```

	npi	nppes_provider_city	nppes_provider_state	nppes_provider_last_org_name	nppes_provider_first_name	specialty_description
0	1003000126	CUMBERLAND	MD	ENKESHAFI	ARDALAN	Internal Medicine
1	1003000126	CUMBERLAND	MD	ENKESHAFI	ARDALAN	Internal Medicine
2	1003000126	CUMBERLAND	MD	ENKESHAFI	ARDALAN	Internal Medicine
3	1003000126	CUMBERLAND	MD	ENKESHAFI	ARDALAN	Internal Medicine
4	1003000126	CUMBERLAND	MD	ENKESHAFI	ARDALAN	Internal Medicine

Ta thấy ở dataset trên có nhiều giá trị bị trùng lặp với nhau . Chúng ta cần loại bỏ đi các giá trị trùng lặp này để data được clean hơn .

```
partD_catfpd = partD_pd0.drop_duplicates()
```

```
partD_catfpd.head()
```

	npi	nppes_provider_city	nppes_provider_state	nppes_provider_last_org_name	nppes_provider_first_name	specialty_description
0	1003000126	CUMBERLAND	MD	ENKESHAFI	ARDALAN	Internal Medicine
16	1003000142	TOLEDO	OH	KHALIL	RASHID	Anesthesiology
38	1003000167	DAYTON	NV	ESCOBAR	JULIO	Dentis
40	1003000282	NASHVILLE	TN	BLAKEMORE	ROSIE	Nurse Practitione
42	1003000407	BROOKVILLE	PA	GIRARDI	DAVID	Family Practice

Sau khi loại bỏ những giá trị trùng lặp bằng hàm drop_duplicates() , chúng ta thu được 1 dataset mới với những giá trị khác nhau .

3. Data Sampling

```
#Split the full dataset into features and targets.
X_full_dataset = full_dataset.drop(["is_fraud"], axis = 1)
Y_full_dataset = full_dataset["is_fraud"].astype(int).copy()

Y_full_dataset
```

```
0          0
1          0
2          0
3          0
4          0
          ..
1287450    0
1287451    0
1287452    0
1287453    0
1287454    0
Name: is_fraud, Length: 1287455, dtype: int64
```

```
Y_full_dataset.unique()
```

```
array([0, 1])
```

```
X_full_dataset.head(3)
```

	npi	specialty_description	total_drug_cost	total_claim	total_day_supply	total_beneficiaries	total_payment_sum	Prscrbr_Crdntl
0	1003000126	Internal Medicine	4.911448	3.050766	4.869707	2.555094	1.752586	M.D
1	1003000142	Anesthesiology	4.577975	3.174351	4.642909	2.477121	1.752586	M.D
2	1003000167	Dentist	2.347642	1.681241	2.658965	1.556303	1.752586	DD

3 rows × 46 columns

Ở đây chúng ta thấy có hơn 1 triệu trường hợp trong dataset .Nhưng chỉ có 304 trường hợp khả thi . Vì vậy nên tổng thể của dataset bị mất cân bằng . Vì vậy nên chúng ta sẽ chọn ra 80000 mẫu ngẫu nhiên cùng với 200 trường hợp khả thi để đưa vào các tập train và test.

```
from imblearn.under_sampling import RandomUnderSampler
# Specify the fixed number of negative instances you want to keep
num_negative_samples = 80000
# Create a RandomUnderSampler
rus = RandomUnderSampler(sampling_strategy = {0: num_negative_samples, 1: 200}, random_state = 41)

# Fit and transform the training data using the RandomUnderSampler
X_resampled_full_dataset, y_resampled_full_dataset = rus.fit_resample(X_full_dataset, Y_full_dataset)
```

```
len(X_resampled_full_dataset)
```

```
80200
```

4. Transforming Data

```
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
```

```
from sklearn.compose import make_column_selector
```

Đầu tiên ta cần import những module từ thư viện scikit-learn để building machine learning pipeline.

Creatng pipeline for numerical features and categorical features of the dataset.

```
num_pipeline = Pipeline([("imputer", SimpleImputer(strategy = "constant", fill_value = 0)),
                         ("std_scaler", StandardScaler())])
```

Biến num_pipeline được tạo ra từ hai bước:

- Điền giá trị thiếu bằng 0 .
- Chuẩn hóa các đặc trưng số bằng StandardScaler

```
cat_pipeline = Pipeline([("one_hot_encoder",
              OneHotEncoder(handle_unknown = "infrequent_if_exist", min_frequency = 10, max_categories = 10))])
```

Biến cat_pipeline sử dụng OneHotEncoder để thực hiện mã hóa one-hot trên các feature của danh sách. Nó xử lý các danh mục không xác định với một giá trị đặc biệt và đặt một tần suất tối thiểu cho danh mục được xét.

```
col_transform = ColumnTransformer(transformers = [
    ("num_pipeline", num_pipeline, all_num_features),
    ("cat_pipeline", cat_pipeline, all_cat_features)], remainder = "drop", sparse_threshold = 0.0)
```

ColumnTransformer sử dụng pipeline cho các column được chỉ định trong all_num_features và các pipeline cho các cột danh sách được chỉ định trong all_cat_features. Tham số remainder="drop" cho biết bất kỳ cột nào không được chỉ định trong các transformer sẽ bị loại bỏ khỏi đầu ra. Tham số sparse_threshold đặt ngưỡng mật độ cho mảng đã được biến đổi; nếu mảng đã biến đổi mật độ hơn ngưỡng này, nó sẽ được chuyển đổi thành ma trận thưa.

```
full_pipeline = Pipeline([("col_transform", col_transform)])
X_train_prepared = full_pipeline.fit_transform(X_train)
```

Cuối cùng chúng ta đóng gói toàn bộ dữ liệu sau quá trình pre-processing vào biến full_pipeline . Toàn bộ pipeline cho dữ liệu X_train được transform theo các định nghĩa trong pipeline sẽ được đưa vào biến X_train_prepared.

Feature Engineering and class balancing

```
#All the numerical features from PartDPrescriberDataset that will be taken as feeatures used for Machine Learning
partD_pd_cat_features = ["Prscrbr_Crdntls", "Prscrbr_Gndr", "Prscrbr_State_Abrvtn", "Prscrbr_RUCA_Desc"]
#All the categorical features from PartDPrescriberDataset that will be taken as feeatures used for Machine Learning
partD_pd_num_features = ["GE65_Tot_Clms", "GE65_Tot_Drug_Cst", "GE65_Tot_Day_Suply", 'GE65_Tot_Benes',
                         "Brnd_Tot_Clms", "Brnd_Tot_Drug_Cst", "Gnrc_Tot_Clms", "Gnrc_Tot_Drug_Cst",
                         "MAPD_Tot_Clms", "MAPD_Tot_Drug_Cst", "PDP_Tot_Clms", "PDP_Tot_Drug_Cst",
                         "LIS_Tot_Clms", "LIS_Drug_Cst", "NonLIS_Tot_Clms", "NonLIS_Drug_Cst",
                         "Opioid_Tot_Clms", "Opioid_Tot_Drug_Cst", "Opioid_Prscrbr_Rate", 'Opioid_Tot_Benes',
                         'Opioid_LA_Tot_Clms', 'Opioid_LA_Tot_Drug_Cst', 'Opioid_LA_Tot_Suply', 'Opioid_LA_Tot_Benes',
                         'Opioid_LA_Prscrbr_Rate',
                         "Antbtc_Tot_Clms", "Antbtc_Tot_Drug_Cst",
                         'Bene_Avg_Age', 'Bene_Age_LT_65_Cnt', 'Bene_Age_65_74_Cnt',
                         'Bene_Age_75_84_Cnt', 'Bene_Age_GT_84_Cnt', 'Bene_Feml_Cnt',
                         'Bene_Male_Cnt', 'Bene_Avg_Risk_Scre']

partD_pd_all_features = partD_pd_cat_features + partD_pd_num_features

#The subset of the original PartDPrescriberDataset that will be used for Machine Learning.
partD_data = partD_pd.loc[:, ["PRSCRBR_NPI"] + partD_pd_all_features]
```

Ở đây có hai danh sách: partD_pd_cat_features cho các đặc trưng phân loại và partD_pd_num_features cho các đặc trưng số. Các danh sách này bao gồm các cột cụ thể từ PartDPrescriberDataset .

Ở đây chúng ta lấy những đặc tính về Credential hành nghề của bác sĩ, giới tinh của bác sĩ, bang mà NPI đó hoạt động. Cùng với đó, những đặc tính về tổng số lần hãng bảo hiểm thanh đoán tiền thuốc, tổng tiền thuốc phải thanh toán hay là tổng số bệnh nhân được hỗ trợ thanh toán tiền thuốc cho những subcategory khác nhau(bệnh nhân trên 65 tuổi, thuốc thuộc loại Brand-name, thuốc thuộc loại Generic, bệnh nhân thu nhập nhấp(Low-income Subsidy), bệnh nhân không phải thu nhập thấp(Non-Low income Subsidy), thuốc phiện(opioid), thuốc phiện lâu dài(long-acting opioid), thuốc kháng sinh(antibiotic)) cũng được thêm vào bộ dữ liệu sau này dùng để huấn luyện. Cùng với đó, nhũng đặc điểm nhân khẩu học của các bệnh nhân đến khám ở từng NPI một cũng được thêm vào bộ dữ liệu(tuổi trung bình, số lượng bệnh nhân theo từng nhóm tuổi, tổng số bệnh nhân chia theo giới tính và điểm rủi ro trung bình) cũng được thêm vào. Những đặc tính này hy vọng sẽ hiệu quả trong quá trình huấn luyện dự đoán NPI gian lận.

Các danh sách đặc trưng phân loại và số tạo ra một danh sách chứa tất cả các đặc trưng có tên là partD_pd_all_features.

Ở đây, một phần của PartDPrescriberDataset gốc được tạo ra. Nó bao gồm cột "PRSCRBR_NPI" và tất cả các đặc trưng được liệt kê trong partD_pd_all_features..

Đoạn mã trên đang tạo ra một tập dữ liệu dùng cho machine learning bằng cách chọn ra các đặc trưng phân loại và số cụ thể từ tập dữ liệu gốc và tạo ra một phần con bao gồm những đặc trưng đó cùng với một cột định danh duy nhất. Phần con này có thể được sử dụng để modelling các mô hình máy học.

```
#all categorical features in the full dataset
all_cat_features = ["specialty_description"] + partD_pd_cat_features
#all numerical features in the full dataset
all_num_features = ["total_drug_cost", "total_claim", "total_day_supply", "total_payment_sum",
                    "total_beneficiaries"] + partD_pd_num_features
```

Đoạn code này xác định hai danh sách: all_cat_features bao gồm một đặc trưng phân loại có tên là "specialty_description" và các đặc trưng phân loại khác từ partD_pd_cat_features. all_num_features bao gồm các đặc trưng số từ bộ dữ liệu đầy đủ, như "total_drug_cost," "total_claim," v.v., và các đặc trưng số bổ sung từ partD_pd_num_features.

```
#scaling of all numerical features
for feature in all_num_features:

    full_dataset[feature] = full_dataset[feature].map(lambda x: np.log10(x + 1.0))
```

Đoạn code này lặp lại qua mỗi đặc trưng số trong all_num_features và áp dụng một biến đổi logarit cho nó. Biến đổi này được sử dụng để chuẩn hóa các giá trị số. Hàm lambda lambda x: np.log10(x + 1.0) lấy logarithm (cơ số 10) của mỗi giá trị trong đặc trưng cộng thêm 1.0 để tránh trường hợp các giá trị = 0 . Đây là một kỹ thuật phổ biến để xử lý các phân phối số có độ lệch chuẩn và cải thiện hiệu suất của các mô hình máy học.

Tóm lại, các đoạn code trên đang chuẩn bị một bộ dữ liệu đầy đủ cho máy học bằng cách xác định các đặc trưng phân loại và số, sau đó chuẩn hóa các đặc trưng số bằng phép biến đổi logarit. Bước tiền xử lý này quan trọng để đảm bảo rằng các đặc trưng có định dạng phù hợp để đào tạo các mô hình máy học.

D. DATA MODELING

- Chúng ta chọn 3 thuật toán để huấn luyện với bộ dữ liệu, đó là Logistic Regression, Random Forest Classifier, Extra Tree Classifier và Gradient Boosting Classifier.

Giới thiệu về từng thuật toán một:

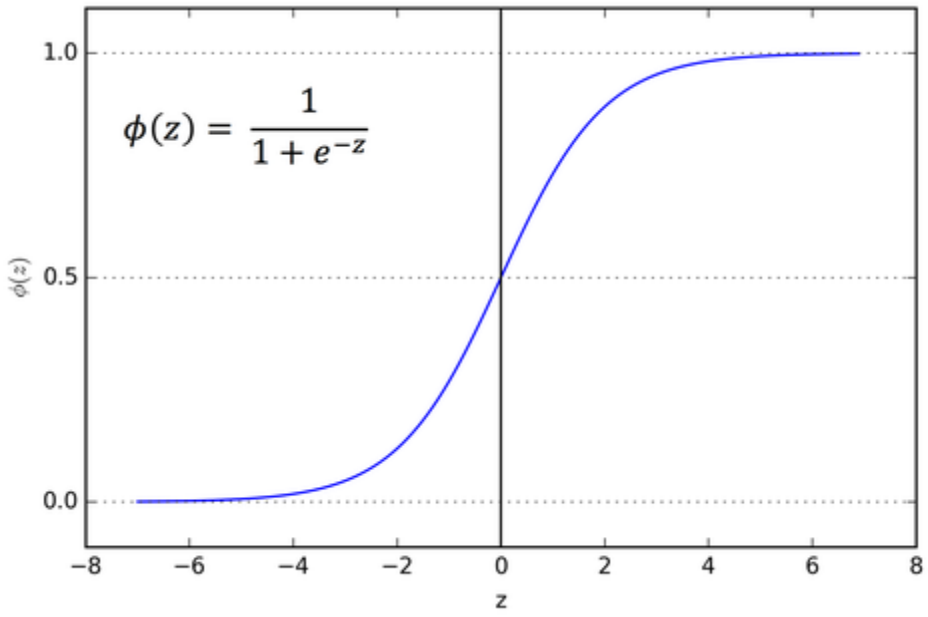
- **Logistic Regression:**

+ Logistic Regression là một thuật toán cơ bản trong Machine Learning được sử dụng cho cả hai loại bài toán Binary Classification(bài toán phân loại mà có 2 category, positive và negative), và Multiclass Classification(bài toán phân loại có nhiều category). Ý tưởng chính đằng sau Logistic Regression là tính xác suất một observation thuộc về một category nhất định(ví dụ thuộc về class label 0 hay 1). Đầu tiên mô hình sẽ tính toán linear combination(dot product) của dữ liệu đầu vào(input features) và tham số của mô hình(model parameters), cộng thêm bias term. Sau đó, mô hình Logistic Regression sẽ áp dụng hàm sigmoid cho kết quả linear combination nói trên. Hàm sigmoid ánh xạ kết quả vào một phạm vi từ 0 đến 1, biểu thị xác suất mà observation thuộc về positive class(trong bài toán Binary Classification). Nếu xác suất này vượt quá ngưỡng đã chọn (thường là 0,5), thì observation đó được phân loại là thuộc positive class; nếu không thì observation đó thuộc về negative class.
+ Trong quá trình huấn luyện, các tham số của mô hình được điều chỉnh lặp đi lặp lại để giảm thiểu cái gọi là loss function, trong mô hình Logistic Regression sẽ là cross-entropy loss, đo lường sự chênh lệch giữa xác suất dự đoán(predicted probability) và nhãn thực tế(actual label). Giống như các thuật toán Machine Learning khác, ác kỹ thuật optimization như Gradient Descent sẽ được sử dụng cho mục đích này. Sau khi được đào tạo, mô hình hồi quy logistic có thể đưa ra dự đoán về dữ liệu mới bằng cách áp dụng các

tham số đã học cho dữ liệu mới chưa từng gặp bao giờ. Thuật toán Logistic Regression rất phổ biến trong các bài toán Classification, và chúng từng được áp dụng rất nhiều trong những ứng dụng thực tế về Recommendation.

$$\mathcal{J}(\mathbf{w}) = -\frac{1}{m}\sum_{i=1}^{m} y_i \log \hat{\mathcal{P}}(1|\mathbf{x}_i, \mathbf{w}) + (1-y_i)\log\left(1-\hat{\mathcal{P}}(0|\mathbf{x}_i, \mathbf{w}\right)$$
$$= -\frac{1}{m}\sum_{i=1}^{m} y_i \log \sigma(\mathbf{w}^T\mathbf{x}_i) + (1-y_i)\log\left(1-\sigma(\mathbf{w}^T\mathbf{x}_i)\right)$$

(Loss function của Logistic Regression, Cross Entropy Loss)



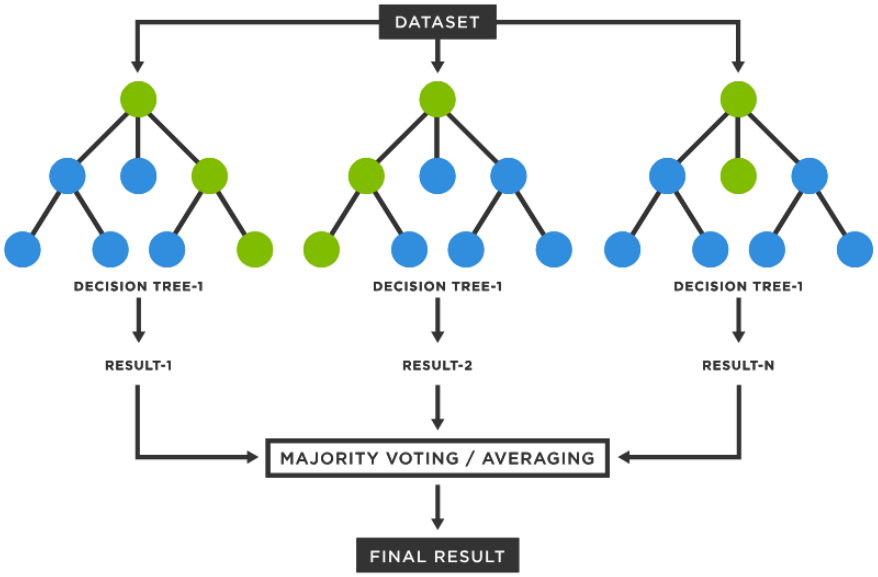
(Sigmoid function)

- **Random Forest:**

+ Random Forest là một thuật toán ensemble được sử dụng rộng rãi cho cả bài toán Classification và Regression trong Machine Learning. Thuật toán xây dựng vô số những cây(số lượng cây được quyết định bởi một hyperparameter) quyết định trong quá trình đào tạo và output ra mode (trong bài toán Classification) hoặc dự đoán trung bình của tất cả các cây (trong bài toán Regression) của từng cây làm dự đoán cuối cùng. Mỗi cây trong Random Forest được huấn luyện trên một tập hợp con dữ liệu ngẫu nhiên và

tại mỗi nhánh(branch) trong mỗi cây, một tập hợp con các features ngẫu nhiên sẽ được xem xét. Điều này mang lại sự đa dạng giữa các cây, vì mỗi cây sẽ được tiếp xúc với những tập hợp con khác nhau của bộ dữ liệu và sử dụng những tập hợp features khác nhau. Điều này giúp giảm thiểu khả năng model bị overfit, nâng cao độ bền vững của mô hình.

+ Trong giai đoạn dự đoán, mỗi cây trong Random Forest sẽ phân loại hoặc dự đoán kết quả một cách độc lập và kết quả cuối cùng của cả Random Forest được xác định bằng cách tổng hợp kết quả của tất cả các cây. Trong bài toán Classification, điều này thường được thực hiện bằng phương thức majority vote, trong đó category xuất hiện thường xuyên nhất trên tất cả các cây sẽ được chọn. Điểm mạnh của Random Forest nằm ở khả năng xử lý dữ liệu nhiều chiều, nắm bắt các pattern phức tạp và cung cấp giải pháp hiệu quả chống lại vấn đề overfitting. Ngoài ra, nó cung cấp chỉ số về tầm quan trọng của những features trong dữ liệu huấn luyện(qua chỉ số feature importance), hỗ trợ việc giải thích quá trình ra quyết định của mô hình.

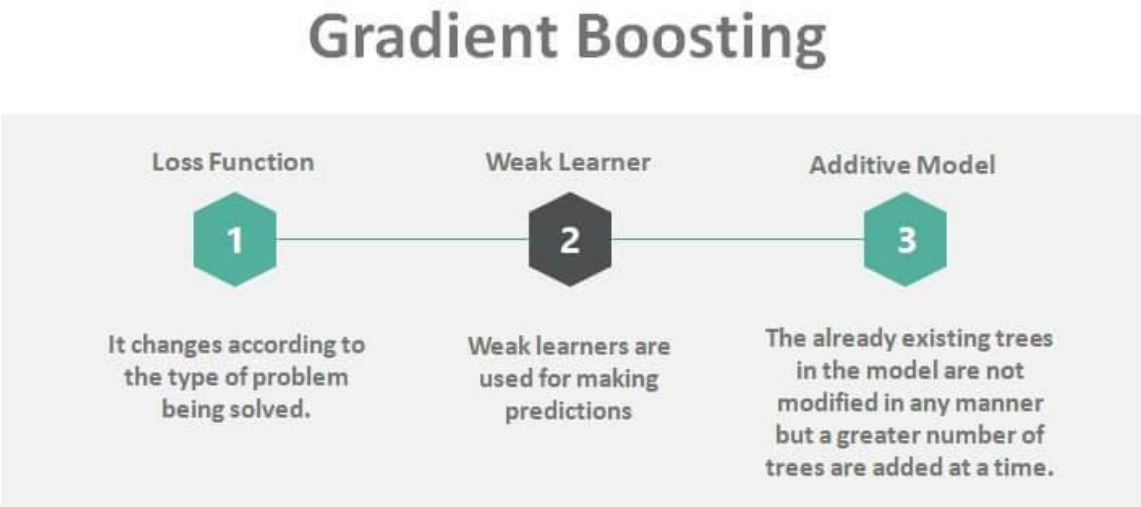


- **Gradient Boosting:**

+ Gradient Boosting là một mô hình Machine Learning theo dạng ensemble nhằm xây dựng một mô hình dự đoán mạnh mẽ bằng cách kết hợp dự đoán của nhiều weak learner(từng cây một), từng cây là một Decision Tree giống như trong mô hình Random Forest. Thuật toán hoạt động theo thứ tự, trong

đó mỗi cây được huấn luyện nhằm mục đích sửa lỗi sai của các cây trước đó. Trong quá trình huấn luyện, mô hình giảm thiểu loss function bằng cách huấn luyện từng cây bằng residual errors của cả ensemble những Decision Tree trước đó. Quá trình huấn luyện này này được chủ hướng bởi gradient của loss function so với dự đoán của mô hình, do đó nó mới có tên là "Gradient Boosting".

+ Trong mỗi lần lặp, một weak learner (thường là một Decision Tree) được thêm vào tập hợp và tham số(parameter) của nó được xác định bằng thuật toán optimization, chẳng hạn như là Gradient Descent. Learning rate là một hyperparameter sẽ kiểm soát sự đóng góp của weak learner vào mô hình tổng thể. Learning rate thấp hơn thường dẫn đến mô hình mạnh mẽ hơn nhưng đòi hỏi nhiều cây hơn để hội tụ. Mô hình Gradient Boosting rất xuất sắc trong việc nắm bắt các mối quan hệ phức tạp trong dữ liệu và ít có xu hướng bị overfit so với các cây quyết định riêng lẻ. Nó được sử dụng rộng rãi cho cả hai bài toán Classification và Regression.



Quá trình huấn luyện mô hình.

- Chúng ta dùng thư viện Scikit-Learn để tạo ra những mô hình Machine Learning để train dữ liệu đang có về những nhà cung cấp dịch vụ chăm sóc sức khỏe (Prescriber, hay là NPI). Đầu tiên chúng ta import những mô hình Machine Learning vào môi trường hiện tại.

Importing Machine Learning training algorithms

In [107]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import GradientBoostingClassifier
```

Creating MachineLearning class

In [108]:
```
log_reg = LogisticRegression(random_state = 42, class_weight = {0: 1, 1: 500}, max_iter = 400, tol = 1e-2)
extra_tree = ExtraTreesClassifier(n_estimators = 300, random_state = 42,
                                  class_weight = {0: 1, 1: 500})
grad_boost = GradientBoostingClassifier(n_estimators = 200, random_state = 42)

classifiers = [log_reg, extra_tree, grad_boost]
```

- Ở đây chúng ta đã import vào môi trường hiện tại 3 mô hình Machine Learning, đó là Logistic Regression, ExtraTreesClassifier và GradientBoostingClassifier. Chúng ta tạo ra các tham số cho những mô hình Machine Learning đó, ví dụ như n_estimators ở hai mô hình Extra Trees Classifier và Gradient Boosting Classifiers là tổng số cây Decision Tree ở trong mô hình, random_state = 42 để đảm bảo rằng các mô hình sẽ ra kết quả giống nhau dù có huấn luyện lại nhiều lần. Trong mô hình Logistic Regression, max_iter thể hiện số bước chạy tối đa của mô hình, mô hình sẽ dùng lại không tiếp tục huấn luyện nữa khi mà hàm loss function đã hội tụ đến giá trị thấp hơn 0.01(vì vậy nên mới có tol = 1e-2). Ở đây, số bước chạy tối đa của mô hình sẽ là 400.

- Sau đó, chúng ta ghép cả 3 mô hình vào 1 list tên là classifier chứa tất cả các mô hình được thử nghiệm trong bài toán Machine Learning này.
- Sau đó, chúng ta huấn luyện từng mô hình một, với input là bộ dữ liệu train đã được preprocess qua scikit-learn pipeline trong phần Preprocessing trước đó, tên là X_train_prepared, với target là y_train. Ở đấy chúng ta sẽ sử dụng kỹ thuật Batch Training(huấn luyện mô hình trước, xong sau đó đưa mô hình vào deploy trên thực tế).

Training each classifier in the list

```
In [109]: for classifier in classifiers:

              classifier.fit(X_train_prepared, y_train)
```

E. REALTIME PREDICT

1. Tổng quan về Kafka

Apache Kafka là một nền tảng truyền tải dữ liệu lớn nguồn mở được sáng tạo phát triển ban đầu bởi công ty LinkedIn. Sau đó, LinkedIn đã chuyện nhượng lại nền tảng Kafka công ty Apache, từ đó Kafka tiếp tục được phát triển và cải tiến bởi Apache. Kafka đã nhanh chóng trở thành một trong những nền tảng dữ liệu lớn hàng đầu trên thế giới. Kafka đã đạt được sự chú ý lớn từ cộng đồng công nghệ thông tin trên toàn thế giới nhờ khả năng xử lý và truyền tải dữ liệu lớn một cách hiệu quả.

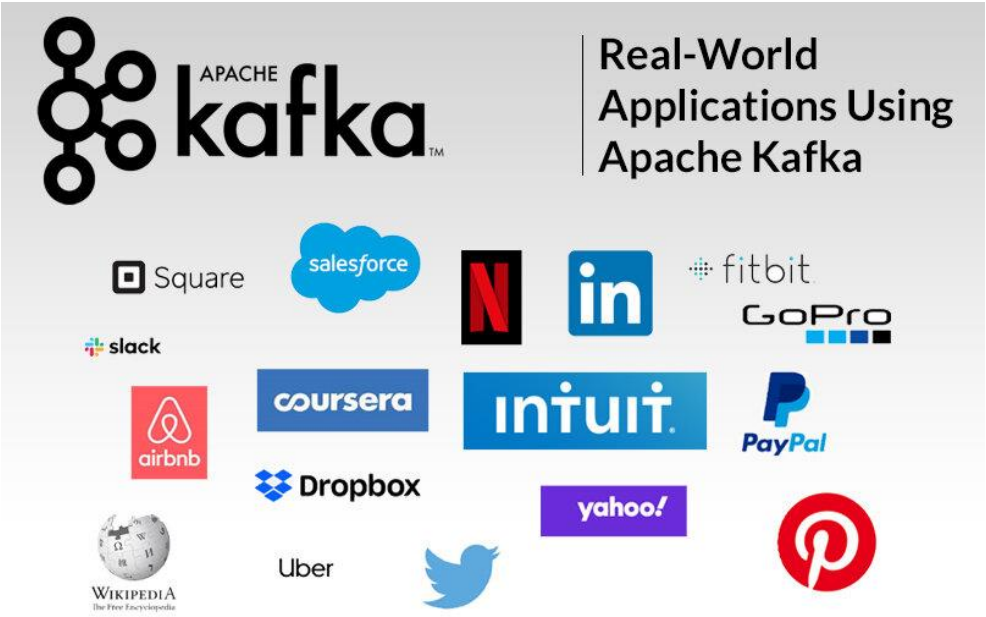
Kafka được biết đến nhiều nhất nhờ hiệu suất tốt, độ trễ(latency) thấp, khả năng chịu lỗi(fault tolerance) và thông lượng cao, Kafka có khả năng xử lý hàng nghìn tin nhắn trong 1 giây. Một số những trường hợp sử dụng phổ biến của Kafka là xây

dựng pipeline để xử lý dữ liệu, tận dụng luồng dữ liệu thời gian thực, kích hoạt các số liệu vận hành(operational metrics) và tích hợp những nguồn dữ liệu khác nhau.

Ngày nay, Kafka được hàng nghìn công ty sử dụng, trong đó có hơn 80% công ty trong danh sách Fortune 100. Trong số đó có Box, Goldman Sachs, Target, Cisco, Intuit, v.v. Là công cụ đáng tin cậy giúp cho các công ty có thể đổi mới và phát triển, Kafka cho phép các công ty có thể hiện đại hóa chiến lược dữ liệu của họ bằng kiến trúc truyền phát dữ liệu hiện đại. Kafka được các công ty hoạt động trong nhiều ngành công nghiệp và dịch vụ sử dụng - từ phần mềm máy tính, dịch vụ tài chính và chăm sóc sức khỏe cho đến việc sử dụng trong các cơ quan chính phủ và ngành giao thông vận tải.

Với khả năng xử lý nhiều tin nhắn trong khoảng thời gian ngắn, Kafka rất phù hợp cho những ứng dụng, hay những công ty có nhu cầu xử lý dữ liệu trong thời gian thực(real-time processing). Khả năng xử lý dữ liệu real-time giúp dữ liệu có thể được xử lý nhanh chóng, những thông tin, dữ liệu quan trọng được nắm bắt gần như ngay lập tức, giúp cho những người ra quyết định trong doanh nghiệp nhanh chóng nắm bắt tình hình và đưa giải pháp.

Kafka được sử dụng bởi các nhà phát triển và kỹ sư dữ liệu tại các công ty như Uber, Square, Strava, Shopify và Spotify.



(Những công ty, tổ chức

đã sử dụng Kafka trong công việc hàng ngày)

- **Những trường hợp sử dụng Kafka phổ biến trong thực tế:**

- **Activity Tracking:**

+ Đây là trường hợp sử dụng ban đầu của Kafka. LinkedIn cần xây dựng lại pipeline theo dõi hoạt động của người dùng dưới dạng tập hợp các pub-sub feed theo thời gian thực. Việc theo dõi hoạt động thường có khối lượng dữ liệu rất cao vì mỗi pageview của user tạo ra nhiều activity message (event):

+ Những sự kiện này có thể được publish vào các Topic riêng của Kafka. Mỗi nguồn cung cấp dữ liệu có sẵn cho bất kỳ trường hợp sử dụng nào, chẳng hạn như tải vào Data Lake hoặc Data Warehouse để xử lý và báo cáo ngoại tuyến.
+ Các ứng dụng khác subscribe Topic, nhận dữ liệu và xử lý dữ liệu khi cần thiết (theo dõi, phân tích, báo cáo, cá nhân hóa trải nghiệm người dùng, v.v.).

- **Real-time Processing:**

+ Nhiều hệ thống yêu cầu dữ liệu phải được xử lý ngay khi có sẵn. Kafka truyền dữ liệu từ Producer đến Consumer với độ trễ rất thấp (ví dụ: 5 mili giây). Điều này hữu ích cho:

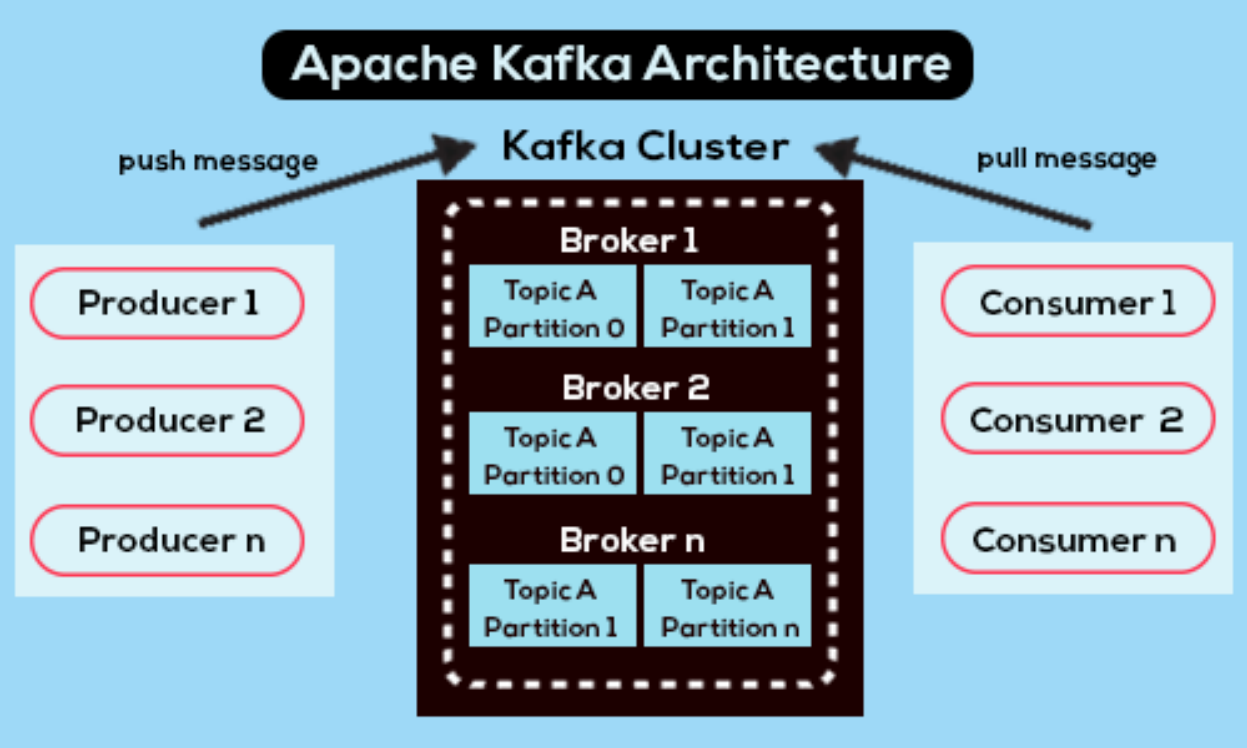
- Các tổ chức tài chính, để thu thập và xử lý các khoản thanh toán và giao dịch tài chính trong thời gian thực, chặn các giao dịch gian lận ngay khi chúng được phát hiện hoặc cập nhật bảng giá chứng khoán real-time với giá thị trường cập nhật đến từng giây.
- Các doanh nghiệp hậu cần và chuỗi cung ứng, để giám sát và cập nhật các ứng dụng theo dõi, chẳng hạn như để theo dõi liên tục các tàu chở hàng để ước tính việc giao hàng theo thời gian thực.

+ Ví dụ về ứng dụng: Một ngân hàng có thể sử dụng Kafka để xử lý các giao dịch trong thời gian thực. Mọi giao dịch do khách hàng thực hiện có thể được publish dưới dạng event cho Kafka Topic. Sau đó, các app có thể sử dụng các event này để xác thực và xử lý các giao dịch, chặn mọi giao dịch đáng ngờ và cập nhật số dư của khách hàng theo thời gian thực.

- **Những khái niệm cơ bản trong Kafka:**

- Kafka chấp nhận các luồng sự kiện**(event)** do nhà sản xuất dữ liệu(data **producer**) viết. Kafka lưu trữ các bản ghi theo thứ tự thời gian trong các phân

vùng**(partition)** trên các **broker (server)**; nhiều nhà môi giới tạo thành một cụm**(cluster)**. Mỗi bản ghi(record) chứa thông tin về một sự kiện và bao gồm một cặp key-value; dấu thời gian và tiêu đề là thông tin bổ sung tùy chọn. Kafka phân loại những bản ghi thành các chủ đề**(Topic)**; người tiêu dùng**(consumer)**nhận được dữ liệu của họ bằng cách subscribe vào các **Topic** trong Kafka.



(Tổng quan kiến trúc của Kafka)

- **Event(Message):** Kafka chấp nhận những luồng event được gửi từ Kafka Producer.

- **Consumers and Producers:**

+ Producer là bất cứ thứ gì tạo ra dữ liệu. Các Producer liên tục viết các event cho Kafka. Ví dụ về Producer bao gồm máy chủ web, các ứng dụng riêng biệt khác (hoặc thành phần ứng dụng), thiết bị IoT, tác nhân giám sát, v.v.

+ Consumer là những thực thể đọc và sử dụng dữ liệu do Producer viết ra. Đôi khi một thực thể có thể vừa là Producer vừa là Consumer; nó phụ thuộc vào kiến trúc hệ thống. Ví dụ: data warehouse có thể sử dụng dữ liệu từ Kafka, sau đó xử lý dữ liệu đó và tạo ra một tập hợp con đã chuẩn bị sẵn để định tuyến lại thông qua

Kafka tới ứng dụng ML hoặc AI. Database, Data Lake và ứng dụng phân tích dữ liệu thường đóng vai trò là Consumer, lưu trữ hoặc phân tích dữ liệu họ nhận được từ Kafka.

- **Broker and Cluster:** Kafka chạy trên các cụm(cluster), mặc dù hiện tại đã có phiên bản Kafka không có máy chủ ở AWS. Mỗi cụm bao gồm nhiều máy chủ(server), thường được gọi là broker (và đôi khi được gọi là node).

+ Đó là điều khiến Kafka trở thành một hệ thống phân tán: dữ liệu trong cụm Kafka được phân phối giữa nhiều broker. Và có nhiều bản sao (replica) của cùng một dữ liệu tồn tại trong một cụm Kafka. Cơ chế này giúp Kafka ổn định hơn, có khả năng chịu lỗi và đáng tin cậy hơn; nếu xảy ra lỗi hoặc lỗi với một server, nhà môi giới khác sẽ thay thế để thực hiện các chức năng của server gặp trục trặc và thông tin sẽ không bị mất.

- **Topic:**

+ Kafka Topic là nhật ký bất biến của các event (trình tự). Producer publish các event(message) lên Topic Kafka; Consumer sẽ subscribe các Topic để truy cập dữ liệu mong muốn. Mỗi Topic có thể phục vụ dữ liệu cho nhiều Consumer, và cũng có thể nhận dữ liệu từ nhiều Producer. Một ví dụ đó là, phần đăng ký tài khoản của trang web xuất bản các event “New User” (thông qua Kafka) vào chủ đề “Registration”, như vậy đó là một Producer. Những Consumer như ứng dụng phân tích, ứng dụng nguồn cấp tin tức, ứng dụng giám sát, Database, v.v. lần lượt sử dụng các event từ chủ đề “Registration” và sử dụng dữ liệu đó với dữ liệu khác làm nền tảng để phân phối sản phẩm hoặc dịch vụ của riêng họ.

- **Partition:**

+ Partition là đơn vị lưu trữ nhỏ nhất trong Kafka. Các phân vùng dùng để phân chia dữ liệu giữa các broker(server) để tăng tốc hiệu suất. Mỗi Kafka Topic được chia thành các partition và mỗi partition có thể được đặt trên một server riêng.

2. Triển khai

- Sau khi đã huấn luyện xong với bộ dữ liệu train, bây giờ chúng ta sẽ sử dụng Kafka, và những metrics khác nhau để đánh giá hiệu quả của những mô hình đã huấn luyện.
- **Cách làm:** Chúng ta tạo ra một Kafka Topic tên là “testing" để nhận những sample của bộ dữ liệu test, sau đó chúng ta sẽ tạo ra một Producer có nhiệm vụ publish dữ liệu test lên Kafka Topic. Sau đó chúng ta tạo ra một Consumer có

nhiệm vụ đọc những message từ Topic “testing" và tính toán những streaming metrics dùng để đánh giá mô hình mỗi khi nhận được 100 sample. Sau khi đã process xong tất cả những sample trên Kafka Topic chúng ta sẽ in ra kết quả để xem mô hình nào có kết quả tốt nhất.

- **Tạo Topic “testing":**

- Tạo topic “testing” bằng câu lệnh sau:

```
(base) nguyentuannguyen@MacBook-Pro-cua-Nguyen kafka_2.13-3.6.0 % ./bin/kafka-topics.sh
--create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic
 testing
Created topic testing.
```

- Bằng câu lệnh trên, chúng ta tạo ra một Kafka Topic tên “testing" với partition là 1, replication factor là 1 và lưu trữ trên localhost server với port 9092.

- Sau đó, chúng ta đưa dữ liệu test qua scikit-learn pipeline mà chúng ta đã xây dựng sẵn trước đó để chuẩn bị đúng dữ liệu test để đưa vào mô hình:

```
X_test_prepared = full_pipeline.transform(X_test)
```

- **Tạo Producer để publish dữ liệu:**

```
from kafka import KafkaProducer
# Kafka broker address, in this case will be locathost:9092
kafka_broker = 'localhost:9092'
# Kafka topic to publish data, in this case will be "training"
testing_kafka_topic = 'testing'

# Create a Kafka producer
testing_producer = KafkaProducer(bootstrap_servers = [kafka_broker],
                         value_serializer = lambda v: json.dumps(v).encode('utf-8'),
                        api_version = (0, 10, 1))
```

- Chúng ta tạo ra Kafka Producer đọc dữ liệu từ broker(server) là localhost:9092(vì dữ liệu của Topic testing nằm trên máy chủ đó), những message sẽ được serialize dưới dạng json khi được publish lên Topic.

- **Publish data lên Topic Testing:**

```
#Sending training instances to the newly created Kafka topic
for i in range(len(X_test_prepared)):

    # Combine random data with actual Iris dataset features
    instance = {
        'features': X_test_prepared[i].tolist(),
        'label': int(y_test.iloc[i])
    }

    # Publish data to Kafka topic
    testing_producer.send(testing_kafka_topic, value = instance)

# Close the Kafka producer
testing_producer.close()
```

- Chúng ta publish dữ liệu lên Topic “testing", với mỗi một sample trong bộ dữ liệu test chúng ta tạo một event dạng từ điển với hai key là “features", nắm giữ giá trị của input, còn “label" là target của sample đó(0 hoặc 1) biểu thị là có gian lận hay không.
- Sau khi đã publish xong tất cả các message, chúng ta đóng Producer lại.

- **Tạo Kafka Consumer:**

```
from kafka import KafkaConsumer
# Create a Kafka consumer
testing_consumer = KafkaConsumer(testing_kafka_topic, bootstrap_servers = [kafka_broker],
                value_deserializer = lambda x: json.loads(x.decode('utf-8')),
                                consumer_timeout_ms = 30000,
                                auto_offset_reset = "earliest")
```

- Tạo Kafka Consumer để subscribe Topic “testing”, đọc dữ liệu từ máy chủ localhost hiện tại. Với mỗi event đọc được thì áp dụng hàm deserialization để decode event đọc được.

- **Đọc dữ liệu từ Topic và áp dụng các metrics đánh giá mô hình:**

- Với mỗi một mô hình Machine Learning được thử nghiệm ở trong bài toán, chúng ta sẽ dùng 3 metrics để đánh giá, đó là Precision, Recall và Area Under The Curve.
- Chúng ta sẽ giải thích sơ qua về ý nghĩa của những metrics này.:

+ Precision đo lường tỷ lệ giữa số lượng các trường hợp dự đoán là đúng (true positives) so với tổng số trường hợp mô hình dự đoán là đúng (bao gồm cả true positives và false positives, trong đó false positives là trường hợp mô hình đoán là đúng nhưng thực ra là sai). Precision đo lường khả năng của mô hình phân loại positive mà không gây ra quá nhiều false positives. Một giá trị precision cao cho thấy mô hình có khả năng cao trong việc xác định chính xác các trường hợp positive. Ngược lại, nếu precision thấp, mô hình có thể đang đánh giá positive cho nhiều trường hợp mà thực sự là negative.
+ Recall được đo lường bằng tỷ lệ giữa số lượng true positives (TP) và tổng số lượng thực tế positive (bao gồm cả true positives and false negatives, trong đó false negatives là những trường hợp dự đoán là sai nhưng thực ra là đúng). Recall tập trung vào khả năng của mô hình trong việc bắt những trường hợp positive thực sự và giữ chúng (ít false negatives). Nó là một metric quan trọng khi chi phí của việc bỏ sót một trường hợp positive là quan trọng, và chúng ta muốn đảm bảo mô hình có khả năng bắt kịp nhiều trường hợp positive nhất có thể.
+ Area Under the Curve (AUC) là một metric được sử dụng để đánh giá độ hiệu quả của mô hình trong bài toán phân loại, đặc biệt là khi đánh giá hiệu suất của mô hình dựa trên đường cong ROC (Receiver Operating Characteristic). Đường cong ROC biểu diễn tỷ lệ True Positive (TP) so với tỷ lệ False Positive (FP) ở nhiều ngưỡng quyết định khác nhau. AUC đo lường diện tích dưới đường cong ROC, và giá trị AUC càng cao thì mô hình càng có hiệu suất tốt. Điều này đồng nghĩa với việc mô hình có khả năng tốt trong việc phân biệt giữa các trường hợp positive và negative ở nhiều mức độ tin cậy khác nhau.

```
features = []
labels = []

#I will use 3 metrics: Precision, Recall and AreaUnderTheCurve to
#evaluate the efficiency of 3 different classifiers
metrics_list = [(classifier, (tf.keras.metrics.Precision(thresholds = 0.02),
                              tf.keras.metrics.Recall(thresholds = 0.02),
                              tf.keras.metrics.AUC()) ) for classifier in classifiers]
# Consume and process data
for message in testing_consumer:
    data_point = message.value

    # Collect features and labels
    features.append(data_point['features'])
    labels.append(data_point['label'])

    if len(features) >= 100:
        for classifier, (precision, recall, auc) in metrics_list:
            sample = np.array(features)
            label = np.array(labels, dtype = np.int16)
            precision.update_state(label,
                                   classifier.predict_proba(sample)[:, 1])
            recall.update_state(label,
                                   classifier.predict_proba(sample)[:, 1])
            auc.update_state(label,
                                   classifier.predict_proba(sample)[:, 1])

        # Reset data lists
        features = []
        labels = []

# Close the Kafka consumer
testing_consumer.close()
```

- Với mỗi classifier chúng ta lập ra những metrics để sau đó tính toán những metrics dựa trên dữ liệu test. Lưu ý rằng ở đây chúng ta sẽ đạt ngưỡng đánh giá xác suất là 0.02, có nghĩa khi mô hình Machine Learning dự đoán xác suất một trường hợp có category là positive, thì nó sẽ ấn định tất cả những trường hợp có xác suất lớn hơn hoặc bằng 0.02 sẽ là positive thay vì 0.5 như bình thường. Ở đây chúng ta phải làm vậy thì tỉ lệ positive/negative trong dữ liệu quá chênh lệch, nên nếu ấn định xác suất là 0.5 thì mô hình sẽ gần như không phát hiện và đánh dấu bất kỳ trường hợp nào là positive(có gian lận) cả.
- Sau đó chúng ta đọc các message đã được gửi lên Topic “testing" bằng Consumer đã tạo, với mỗi 100 trường hợp thì tính toán và update metrics với tất cả 3 classifier, cuối cùng đóng Kafka Consumer lại.

RESULT

- Kết quả cuối cùng của 3 mô hình lần lượt như sau:

```
LogisticRegression

The precision score on the test set is: 0.0027059864
The recall score on the test set is: 0.98
The roc auc score on the test set is: 0.8011905

RandomForestClassifier

The precision score on the test set is: 0.01724138
The recall score on the test set is: 0.12
The roc auc score on the test set is: 0.6562482

GradientBoostingClassifier

The precision score on the test set is: 0.0
The recall score on the test set is: 0.0
The roc auc score on the test set is: 0.52317137
```

- Dựa vào kết quả nói trên, chúng ta có những đánh giá như sau:

+ Mô hình Logistic Regression có ưu điểm là tỉ lệ Recall cao, có nghĩa là mô hình có thể bắt được tới 98% những trường hợp, những nhà cung cấp dịch vụ y tế(NPI) thực sự có gian lận trong hoạt động. Tuy nhiên, tỉ lệ Precision của mô hình là khá thấp, chỉ có 0.2%. Điều này có nghĩa là khi mô hình đánh giá một NPI là gian lận, thì tỉ lệ phần trăm mà NPI đó thực sự gian lận chỉ là 0.2%, còn lại là trong sach(mô hình phán đoán sai). Có nghĩa là mô hình sẽ bắt oan khá nhiều nhà cung cấp mà thực ra không có gian lận gì trong hoạt động. Tỉ lệ AUC là 80%, có nghĩa là mô hình có khả năng tương đối tốt trong việc phân biệt giữa các trường hợp positive và negative ở nhiều mức độ tin cậy khác nhau.
+ Mô hình Random Forest Classifier có tỉ lệ Recall thấp hơn, có nghĩa là mô hình chỉ bắt được tổng cộng 12% những trường hợp gian lận. Nhưng bù lại khi mô hình đánh giá một trường hợp gian lận thì tỉ lệ chính xác sẽ cao hơn Logistic Regression(1%), như vậy mức độ rủi ro oan sai được giảm đi(tuy vẫn khá cao). Tỉ lệ AUC là 65%, có nghĩa khả năng của mô hình trong việc phân biệt trường hợp positive và negative kém hơn của Logistic Regression.
+ Mô hình Gradient Boosting Classifier có điểm đánh giá quá kém nên chúng ta sẽ không bàn nhiều ở đây.

- **Kết luận:**

+ Chúng ta sẽ so sánh giữa hai mô hình Logistic Regression và Random Forest. Hai mô hình có điểm yếu và điểm mạnh khác nhau(Logistic Regression có tỉ lệ Recall cao nhưng Precision thấp hơn, còn Random Forest có tỉ lệ Recall thấp hơn nhưng Precision cao hơn). Mô hình Logistic Regression sẽ bắt được nhiều trường hợp thực sự gian lận, những cũng sẽ bắt nhiều trường hợp vô tội, trong khi Random Forest sẽ bắt được ít trường hợp gian lận hơn nhưng tỉ lệ bắt oan ít hơn.
+ Trong bài toán về Fraud Detection trong Medicare, có thể sẽ có khả năng tỉ lệ Precision sẽ được ưu tiên hơn, vì người làm chính sách sẽ ưu tiên không bắt oan hay gây phiền toái cho những chủ thể vô tội, cho dù sẽ có khả năng để lọt nhiều trường hợp gian lận, nếu như vậy thì mô hình Random Forest sẽ được ưu tiên nhiều hơn.
+ Cả hai mô hình trên đều có tỉ lệ Precision khá kém, có nghĩa là khi mô hình phát hiện ra trường hợp gian lận thì dự đoán đó sẽ không thực sự quá đáng tin, như vậy sẽ gây phiền hà cho nhiều nhà cung cấp làm việc theo đúng pháp luật. Như vậy thì thử nghiệm với những mô hình trên có thể đã không quá thành công và chưa thể đem vào áp dụng trên quy mô lớn. Có thể sẽ cần thêm những đặc tính mới cho từng NPI, cộng thêm những mô hình tiên tiến hơn hay cách preprocess dữ liệu tốt hơn để cải thiện Precision cho mô hình trong khi vẫn giữ cho Recall tương đối cao. Những mô hình Machine Learning về phát hiện gian lận trong lĩnh vực chăm sóc sức khoẻ nên được cân nhắc kỹ trước khi đưa vào áp dụng quy mô lớn, và nên được áp dụng kèm thêm những biện pháp kiểm tra khác nhằm phát hiện và theo dõi trường hợp gian lận.

CONCLUSION

Tóm lại, trong bài báo cáo này, chúng tôi đã thành công xây dựng một ứng dụng giúp nhận diện các gian lận theo thời gian thực với hiệu suất tạm ổn. Kết quả nhận diện gian lận tuy chưa thực sự tốt nhưng nó sẽ là một công cụ tốt trong việc hỗ trợ việc phòng chống các gian lận tài chính trong y tế như giảm nhân lực điều tra, khoanh vùng những đối tượng có khả năng gian lận cao.

REFERENCE

Part D Prescriber Dataset: https://data.cms.gov/provider-summary-by-type-of-service/medicare-part-d-prescribers/medicare-part-d-prescribers-by-provider

Payment Received dataset Dataset Link: https://www.cms.gov/priorities/key-initiatives/open-payments/data/dataset-downloads

Excluded (LEIE) dataset: https://oig.hhs.gov/exclusions/exclusions_list.asp

What is Random Forest: https://www.ibm.com/topics/random-forest#:~:text=Random%20forest%20is%20a%20commonly,both%20classification%20and%20regression%20problems.

What is Logistic Regression: https://www.ibm.com/topics/logistic-regression

Apache Kafka Use Cases, When To Use it? When Not To?: https://www.upsolver.com/blog/apache-kafka-use-cases-when-to-use-not