UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E COMPUTAÇÃO CIENTÍFICA
DEPARTAMENTO DE MATEMÁTICA APLICADA

SABRINA BONFIM BARBOSA

Alguns estudos preliminares referentes a previsão de customer churn usando sistemas fuzzy

Campinas
2022

SABRINA BONFIM BARBOSA

Alguns estudos preliminares referentes a previsão de customer churn usando sistemas fuzzy

Monografia apresentada ao Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas como parte dos requisitos para obtenção de créditos na disciplina Projeto Supervisionado, sob a orientação do Prof. Peter Sussner.

Resumo

Nos últimos anos, *Customer Relationship Management* (CRM) tornou-se uma ferramenta essencial para muitas empresas que buscam reduzir a perda de clientes, já que esta pode trazer impactos significativos na lucratividade e vantagens competitivas de uma companhia. Portanto, redução e previsão de *customer churn* é uma das estratégias mais importantes de CRM. Existem muitos métodos de *machine learning* adequados para lidar com a previsão de *customer churn*, visto que estes geralmente necessitam de tratar de grande volume de dados. Sendo assim, muitas imprecisões e incertezas podem ser encontradas quando lidamos com bancos de dados com muitos parâmetros diversos. Por outro lado, a lógica fuzzy nos permite lidar com dados numéricos e, ainda assim, ser aplicável para variáveis de entrada não-numéricas e linguísticas.

Abstract

In recent years, Customer Relationship Management (CRM) has become essential for many business organizations looking to decrease customer defections, as it can have a significant impact on a business' profit and competitive advantage in its market. Therefore, customer churn prediction and prevention is one of CRM's most important strategies. There are many machine learning methods that are applicable to solve customer churn problems, since it deals with a great amount of data, but many imprecisions and uncertainty can exist while dealing with diverse customer datasets. In other hand, fuzzy logic is all about dealing with numerical data while being suitable to non-numerical and linguistic variables inputs.

Conteúdo

1 Introdução

Customer Relationship Management (CRM), em tradução livre "Gestão de Relacionamento com Cliente", é um modelo de estratégia de negócios que utiliza tecnologia da informação (TI) para melhorar as relações empresa-cliente, de forma que estas sejam duradouras e lucrativas, pois o custo de retenção de clientes é muito menor do que o custo de aquisição de novos clientes. Com isso, a previsão de *customer churn* se tornou uma ferramenta essencial para muitas empresas. Existem diversos métodos de *machine learning* que podem ser usados para resolver o problema de previsão de *customer churn*, como redes neurais, árvores de decisões e regressão logística. Independente de qual o método utilizado, para o sucesso do modelo, é necessário conhecimento especialista da base de clientes que se deseja estudar.

Neste contexto, existem diversos parâmetros que podem ser usados em modelo de previsão de *customer churn*, como tempo de adesão do cliente a companhia ou quantidade mensal de compras, para definir como determinado cliente pode ser uma deserção. Diante disso, incertezas surgem ao lidar com tais parâmetros e perguntas sobre como e com qual intensidade estes devem ser avaliados surgem.

Durante um grande período da história, a incerteza foi vista como indesejável. A visão mais tradicionalista da ciência assumia que esta deveria buscar a certeza em todas as suas formas. A mudança de paradigma começa a ocorrer no final do século 19, com os estudos em Física avançando em nível molecular [Yuan et al, 1995]. Em uma visão moderna sobre a incerteza, esta se torna uma parte essencial para a ciência, com suas aplicações nas mais diversas áreas, como economia, computação e negócios.

Nesta monografia, iremos estudar a aplicação de um modelo de inferência fuzzy para resolver o problema de previsão de *customer churn* de uma base de dados de clientes de um *marketplace* hipotético.

2 O problema de *Customer churn*

O *customer churn*, também chamado de *customer attrition*, se refere à tendência natural de clientes deixarem de fazer negócios com uma determinada empresa ou dos funcionários de uma organização se desligarem. Existem diversos fatores que podem levar a

ocorrência de um *churn* e, entendê-los e buscar soluções para evitá-lo é fundamental para uma empresa manter a rentabilidade de seu negócio.

Um exemplo bastante ilustrativo seria o de um serviço de *streaming*, como a Netflix ou Amazon Prime Video. No caso desses serviços, o *churn* ocorre quando um cliente cancela sua assinatura voluntariamente ou tem sua assinatura cancelada automaticamente pela plataforma.

2.1 *Churn rate*

O *churn rate*, ou *attrition rate*, é a métrica que é indica o percentual dos clientes que sofrem *churn* em um determinado período de tempo, que é definido pelas regras internas dos times de negócios de cada instituição. É calculada como

$$Churn\ rate = \frac{(Qtd.\ de\ clientes\ no\ INÍCIO\ do\ período - Qtd.\ de\ clientes\ no\ FINAL\ do\ período)}{(Qtd.\ de\ clientes\ no\ INÍCIO\ do\ período)}$$

Evidentemente, quanto menor for o *churn* de uma empresa, mais clientes esta mantém.

2.2 Por que prever o *customer churn*?

A saída de clientes de uma companhia pode impactar muito essa, afetando desde sua lucratividade até sua vantagem competitiva no mercado na qual está inserida, isto porque a relação cliente-companhia é um resultado direto da qualidade de serviços que esta última pode oferecer.

É intuitivo imaginar que adquirir novos clientes seja mais custoso do que manter os que já possuem vínculos com a companhia, pois, para atrair novos clientes é geralmente necessário promoções, investimento em marketing e publicidade, entre outras técnicas de negócios, além dos custos de operação que envolvem integrar novos parceiros aos sistemas da empresa. Enquanto clientes que já estão há um tempo na companhia já não possuem mais esses mesmo custo e, além disso, são potenciais fontes de rentabilidade. Por já estarem há um determinado período com a companhia, esta última também aprende a melhorar seus serviços para atendê-los de forma mais eficiente, podendo aumentar sua

lucratividade.

Quando uma empresa opta por encontrar formas para identificar o *churn*, ela também está buscando pontos fracos em sua estrutura que levam seus clientes a escolherem encerrar suas parcerias e, também, está buscando soluções para esses pontos fracos. Prever o *customer churn* é, essencialmente, prevenir *customer churn* e, consequentemente aumentar a retenção de seus parceiros.

3 Algumas Noções de Lógica Fuzzy

3.1 Teoria dos Conjuntos Fuzzy

A Teoria dos Conjuntos Fuzzy foi introduzida em 1965, por Lotfi Asker Zadeh, com o objetivo de formalizar matematicamente termos linguísticos associados à incerteza e imprecisão.

Dado um conjunto clássico U, com subconjunto A, este pode ser definido por sua função característica, da forma,

$$\chi_A(x) = \begin{cases} 1, \ se \ x \in A \\ 0, \ se \ x \notin A \end{cases}$$

Porém, existem casos em que definir se um elemento pertence ou não a um determinado conjunto não é exatamente precisa, ou seja, não é possível definir o quanto determinado elemento pertence ou não ao conjunto.

Por exemplo, vamos considerar o conjunto P dos números naturais pequeno, assim, se n é pequeno, então $n \in P$. No entanto, o que seria dizer que um número n é pequeno? Quais valores de n poderiam ser considerados pequenos? Não é trivial responder essa questão de forma binária. Temos a seguinte definição:

Seja U um conjunto clássico; temos que um subconjunto fuzzy F de U é caracterizado pela sua função, chamada de função de pertinência, que indica o grau com que o elemento x de U está no conjunto fuzzy F:

$$\varphi_F : U \to [0, 1]$$

$$\varphi_F \in [0, 1]$$

Um subconjunto fuzzy F é composto de elementos x de um conjunto clássico U, providos de um valor de pertinência de F, dado por φ_F. Assim, podemos dizer que um subconjunto fuzzy F de U é dado por um conjunto clássico de pares ordenados:

$$F = \{(x,\ \varphi_F),\ com\ x \in U\}$$

3.2 Lógica Fuzzy

Nesta seção, iremos nos referir a lógica fuzzy como uma extensão da lógica clássica. Quando iniciamos o aprendizado em lógica matemática, estudamos os conectivos "e" e "ou", estes são essenciais para o entendimento da implicação de uma sentença lógica. Estes conectivos são estudados por meio de tabelas verdade Dessa forma, o valor lógico de uma sentença, formada a partir de duas ou mais proposições, é obtido por meio de composições das tabelas verdades dos conectivos presentes nesta sentença [Barros et al, 2010].

Na lógica clássica, as sentenças lógicas só podem assumir dois valores: 1, quando são verdadeiras, ou 0, quando falsas.

p	q	p $\wedge$ q
1	1	1
1	0	0
0	1	0
0	0	0

Tabela 1: Tabela verdade de $\wedge$ (mínimo). Fonte: [Barros et al, 2010]

p	q	p $\vee$ q
1	1	1
1	0	1
0	1	1
0	0	0

Tabela 2: Tabela verdade de $\vee$ (máximo). Fonte: [Barros et al, 2010]

p	$\neg$ p
1	0
0	1

Tabela 3: Tabela verdade de $\neg$ (negação). Fonte: [Barros et al, 2010]

p	q	p $\Rightarrow$ q
1	1	1
1	0	0
0	1	1
0	0	1

Tabela 4: Tabela verdade de $\Rightarrow$ (implicação). Fonte: [Barros et al, 2010]

Quando extendemos essa tabela verdade para a lógica fuzzy, podemos assumir a operação mínimo para o conectivo "e", assim, dados os conjuntos, A e B, dentro do

domínio (1,0), usaremos o *min(A, B)*. Analogamente, para o conectivo "ou", usaremos *max(A,B)*. Já em relação a negação, estabelecemos: $\neg A \Rightarrow 1 - A$.

3.2.1 Operadores Lógicos

Uma *t-norma* é um operador $\triangle[0,1] \times [0,1] \rightarrow [0,1],\ \triangle(x,y) = x \triangle y$, que é uma extensão do operador *mínimo* ($\wedge$), modelando o conectivo lógico "e". A t-norma satisfaz as seguintes propriedades:

1. elemento neutro: $\triangle(1,x) = 1 \triangle x = x$;
2. comutativa: $\triangle(x,y) = x \triangle y = y \triangle x = \triangle(y,x)$;
3. associativa: $x \triangle (y \triangle z) = (x \triangle y) \triangle z$;
4. monotonicidade: *se* $x \leq u$ *e* $y \leq v$, *então* $x \triangle y \leq u \triangle v$.

De forma similar, para o conectivo lógico "ou", vimos anteriormente que este é modelado pela operação *máximo* ($\vee$), temos o operador *t-conorma* $\bigtriangledown(x,y) = x \bigtriangledown y$, que satisfaz as condições:

1. elemento neutro: $\bigtriangledown(0,x) = 0 \bigtriangledown x = x$;
2. comutativa: $\bigtriangledown(x,y) = x \bigtriangledown y = y \bigtriangledown x = \bigtriangledown(y,x)$;
3. associativa: $x \bigtriangledown (y \bigtriangledown z) = (x \bigtriangledown y) \bigtriangledown z$;
4. monotonicidade: *se* $x \leq u$ *e* $y \leq v$, *então* $x \bigtriangledown y \leq u \bigtriangledown v$.

Uma implicação fuzzy é uma aplicação $[0,1] \times [0,1] \rightarrow [0,1]$ que satisfaz as seguintes condições:

1. Reproduz a tabela clássica de lógica;
2. O operador é decrescente no primeiro argumento e crescente no segundo argumento.

3.3 Base de Regras Fuzzy

Uma regra fuzzy é uma proposição da forma *If-Then* ("Se-Então", em inglês),

Se "*estado*", então "*resposta*"

na qual "*estado*" e "*resposta*" são valores assumidos por variáveis linguísticas que serão, consequentemente, modelados por conjuntos fuzzy. Uma base de regras fuzzy é um conjunto de regras que "traduz" matematicamente esses termos linguísticos [Barros et al, 2010].

3.4 Sistemas Baseados em Regras Fuzzy (SBRF)

O Sistema Baseado em Regras Fuzzy (SBRF) é um sistema que utiliza lógica fuzzy para construir *outputs* (saídas) para cada *input* (entrada) fuzzy. [Barros 2006]. Um caso particular de um SBRF é o de um controlador fuzzy, no qual cada *output* representa a "ação" correspondente à "condição", definida pelo *input* do SBRF. Sendo assim, uma regra fuzzy de um controlador fuzzy é definida como,

Se *"ação"*, então *"condição"*

Um controlador fuzzy pode ser construído utilizando os seguintes passos:

1. Fuzzificação: os *inputs* são modelados conjuntos fuzzy em seus domínios e as funções de pertinência são formuladas. Nesta etapa, podemos ter *inputs crisp*, que serão "fuzzificados" conforme suas funções características;

2. Base de Regras: etapa na qual as proposições fuzzy (*If-Then*) são classificadas de forma linguística e, em seguida, são modeladas suas funções de pertinências;

3. Inferência Fuzzy: módulo em que a base de regras é interpretada matematicamente por meio da lógica fuzzy;

4. Defuzzificação: é o módulo em que se representa o conjunto fuzzy por valores *crisp*.

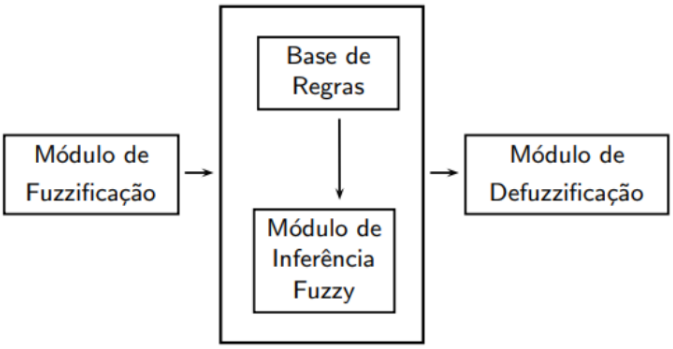

Figura 1: Esquema de um controlador fuzzy. Fonte: [Barros et al, 2010]

4 Aplicações

Vamos criar um modelo de previsão de *customer churn*, com base na situação hipotética dos vendedores de um *marketplace online*. Para isso, utilizaremos a linguagem Python e as bibliotecas *NumPy*, *Pandas*, *datetime*, *matplotlib* e *skfuzzy*, sendo esta última especificamente para lidar com problemas fuzzy. Para testar o modelo, será necessário criar um banco de dados, com dados aleatórios.

4.1 *Dataset*

Para aplicar o modelo fuzzy, foi criado um *dataset* com mil linhas diferentes, representando os vendedores da plataforma, utilizando as seguintes bibliotecas do Python:

1. *Pandas*: para criação do *dataset*;
2. *random*: escolher valores aleatórias dentro do *dataset*;
3. *uuid*: gerar diferentes *ids* para a granularidade única da base de dados;
4. *NumPy*: biblioteca padrão do Python.

A base de dados contém quatro colunas, sendo uma de *id* para identificar cada vendedor como único dentro do *dataset*, e outras três métricas que serão significativas para o nosso modelo. São elas:

- Quantidade de estoque: é a quantidade de estoque que o vendedor possui cadastrado, um valor inteiro (tipo *int*) e pode variar de 0 a 500;

- Data da última venda: data da última venda do vendedor dentro do *marketplace*, um valor de data do período de um ano, de 01/10/2021 a 01/10/2022;

- Data do último cadastro de estoque: data do último cadastro de estoque dentro da plataforma, também é um valor de data do período de um ano, de 01/10/2021 a 01/10/2022.

Todas as métricas precisam estar em valores numéricos para o modelo, assim, será necessário converter as datas para valores numéricos. A biblioteca *datetime*, do Python, possui a função *.timestamp()*, que transforma uma data em um *timestamp*. Um *timestamp* é uma informação codificada de um registro da data e hora no qual um evento particular ocorreu, e é facilmente convertida para *int*.

4.2 Base de Regras

A base de regras do nosso modelo utilizará as três métricas que foram geradas para o nosso *dataset* como os *inputs* (variáveis de entrada), e um *output* (variável de saída), que será chamado de *churn*.

4.2.1 *Output*

A variável de saída, o *churn*, tem três resultados possíveis, sendo modelados pelas seguintes funções triangulares:

1. Sem Churn;

$$\mu_{ZERO} = \begin{cases} 1, \ x \leq 0 \\ \frac{x}{0.6}, \ 0 \leq x \leq 0.6 \\ 0, \ x \geq 0.6 \end{cases}$$

2. Churn A, quando há a possibilidade do vendedor ser reativado na plataforma;

$$\mu_{A} = \begin{cases} 0, \ x \leq 0 \\ \frac{x}{0.6}, \ 0 \leq x \leq 0.6 \\ \frac{1-x}{0.4}, \ 0.6 \leq x \leq 1 \\ 0, x \geq 1 \end{cases}$$

3. Churn D, quando o vendedor é desativado permanentemente da plataforma.

$$\mu_D = \begin{cases} 0, x \leq 0.6 \\ \frac{1-x}{0.6}, 0.6 \leq x \leq 1 \\ 1, x \geq 1 \end{cases}$$

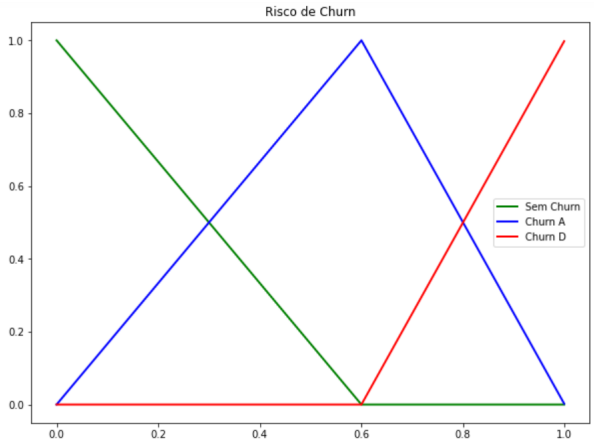

Figura 2: Gráfico da função de pertinência do consequente *churn*

4.2.2 *Input*

Para todas as métricas, todas serão definidas conforme variáveis linguísticas, sendo classificadas em "Baixo", "Médio" e "Alto".

Para o *input* de quantidade de estoque, temos valores que vão de 0 a 500, enquanto as variáveis de datas vão de 1633057200 a 1664593200,que são os valores das datas 01/10/2021 e 01/10/2022, respectivamente, convertidos para *timestamp*. Assim, temos as seguintes funções de pertinência triangulares.

Quantidade de estoque:

1. Baixo: $\mu_{baixo} = \begin{cases} 1, & x \leq 0 \\ \frac{x}{250}, & 0 \leq x \leq 250 \\ 0, & x \geq 250 \end{cases}$

2. Médio: $\mu_{médio} = \begin{cases} 0, & x \leq 0 \\ \frac{x}{250}, & 0 \leq x \leq 250 \\ \frac{500-x}{250}, & 250 \leq x \leq 500 \\ 0, x \geq 500 \end{cases}$

$$3.\ \text{Alto: } \mu_{alto} = \begin{cases} 0, & x \leq 250 \\ \frac{500-x}{250}, & 250 \leq x \leq 500 \\ 1, & x \geq 500 \end{cases}$$

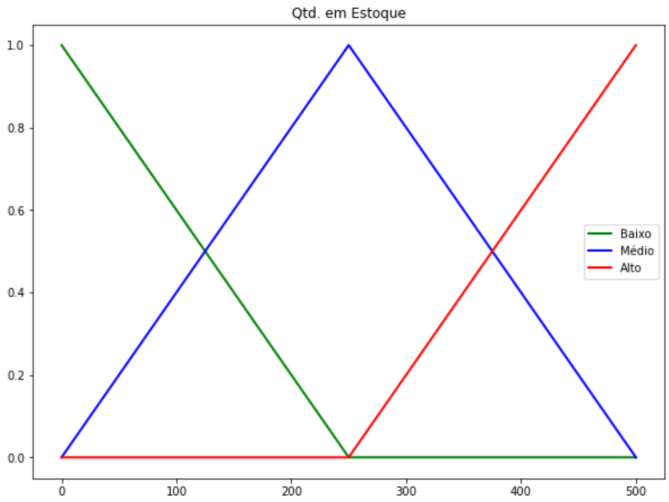

Figura 3: Gráfico da função de pertinência do antecedente *quantidade de estoque*

Datas de Último Cadastro de Estoque e Última Venda:

$$1.\ \text{Baixo: } \mu_{baixo} = \begin{cases} 1, & x \leq 1633057200 \\ \frac{x-1633057200}{15724800}, & 1633057200 \leq x \leq 1648782000 \\ 0, & x \geq 1648782000 \end{cases}$$

$$2.\ \text{Médio: } \mu_{medio} = \begin{cases} 0, & x \leq 1633057200 \\ \frac{x-1633057200}{15724800}, & 1633057200 \leq x \leq 1648782000 \\ \frac{1-x}{15811200}, & 1648782000 \leq x \leq 1664593200 \\ 0, x \geq 1664593200 \end{cases}$$

$$3.\ \text{Alto: } \mu_{alto} = \begin{cases} 0, & x \leq 1648782000 \\ \frac{1-x}{15811200}, & 1648782000 \leq x \leq 16645932001 \\ 1, & x \geq 16645932001 \end{cases}$$

4.2.3 Base de Regras

Nosso sistema possui uma estrutura do tipo MISO (*Multiple Input Single Output*) e contém 25 regras. Todas as regras são dependentes das três variáveis de entrada

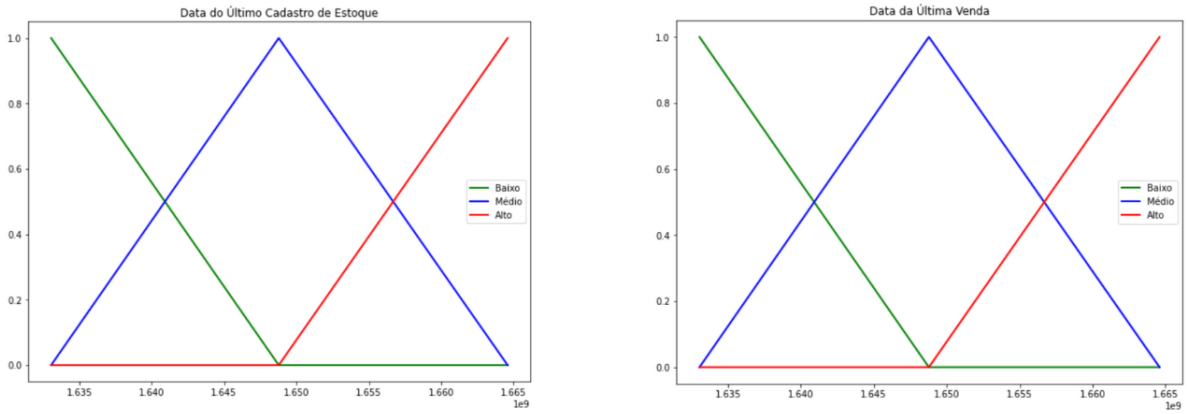

Figura 4: Gráficos das funções de pertinência dos antecedentes *data do último cadastro de estoque* e *data da última venda*

assumirem um estado, sendo assim, as funções serão do tipo *min(input 1, input 2, input 3)*.

4.3 Método de Inferência de Mamdani

O sistema de inferência escolhido para construirmos o controlador fuzzy foi o Método de Inferência de Mamdani, particularmente, por ser mais intuitivo e pela facilidade em interpretar bases de regras formuladas por um modelo de expertise. Este método é baseado na regra de composição de inferência *max–min*, da seguinte forma:

1. Cada regra, da nossa base de regras previamente formulada, será modelada pela função de mínimo ($\wedge$);

2. Adotamos a t-norma $\wedge$ (mínimo) para o conectivo lógico "e" e, para o conectivo "ou", adotamos a t-conorma $\vee$ (máximo).

Neste passo, utilizamos as funções da biblioteca NumPy, *fmin()* e *fmax()*. A função *fmin()*, que foi aplicada às 25 regras da nossa base de regras, compara dois *arrays* e retorna o elemento mínimo encontrado entre eles, ignorando valores *NaN* (*Not a Number*, um tipo de dado que é indefinido pelo compilador). A função *fmax()*, assim como a função mínima, compara dois *arrays*, retornando o elemento máximo e ignorando valores *NaN*.

Para visualizar o controlador fuzzy, vamos olhar algumas regras ativadas de um vendedor aleatório dentro do nosso *dataset*, que chamaremos de vendedor 1. Este possui quantidade de estoque igual a 15, data de último cadastro é 12/06/2022 e data da última

venda igual a 05/07/2022 que, convertendo para *timestamp*, são iguais a 1655002800 e 1656990000, respectivamente. Para este vendedor, a primeira regra a ser ativada foi a regra 4, da nossa base de dados, e a regra 25 foi a última regra ativada. O *output* obtido, defuzzificado pelo método do centróide, é igual a 0.565.

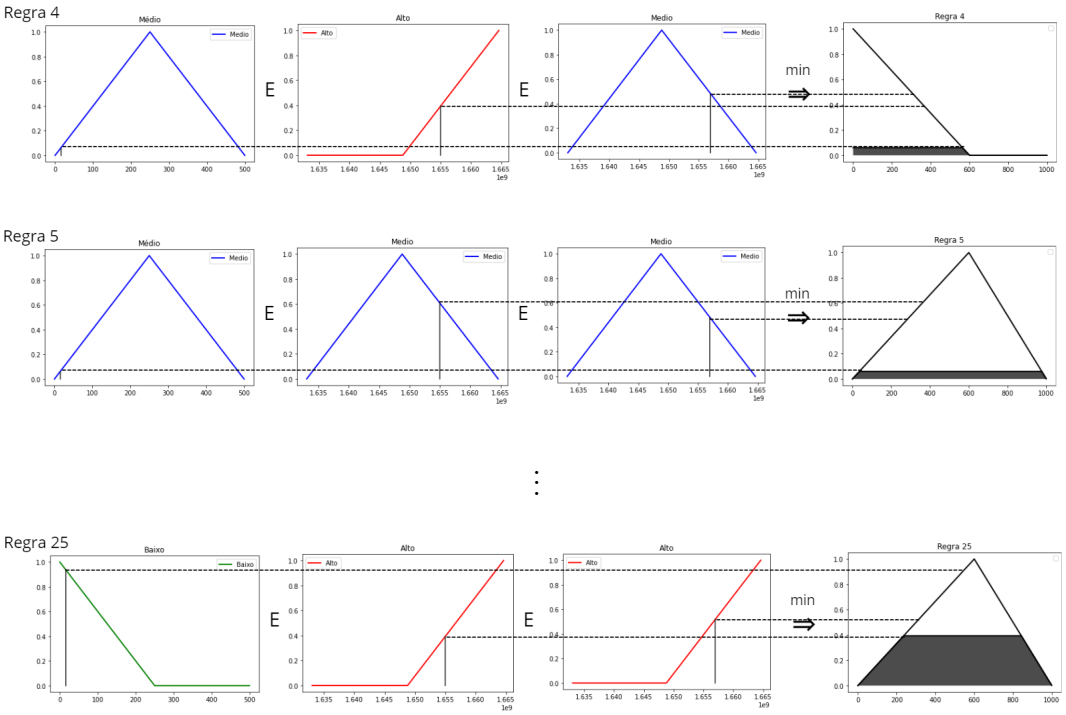

Figura 5: Saídas das regras ativadas do controlador fuzzy de Mamdani para o vendedor 1

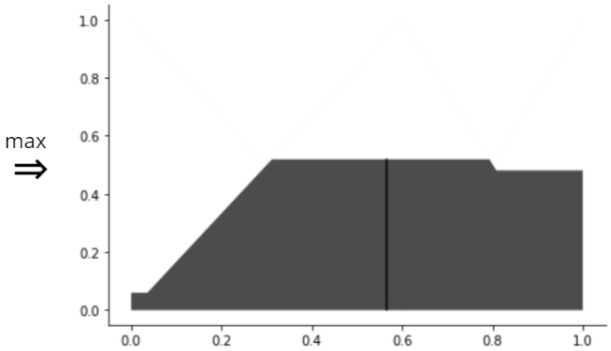

Figura 6: Saída final do controlador fuzzy de Mamdani para o vendedor 1, com o *output* defuzzicado

4.4 Defuzzificação

Em um controlador fuzzy, mesmo que tenhamos entradas *crisp*, as saídas ainda serão do tipo fuzzy. Portanto, torna-se necessário "defuzzificar" essa saída e existem alguns métodos para isso, já que é extensa a variedade de métodos de defuzzificação.

No nosso modelo, usaremos o método de Centróide, cuja ideia assemelha-se à média aritmética da distribuição de frequências de uma certa variável. Aqui, o centro de gravidade da média das áreas de todas as figuras que representam os graus de pertinência de um subconjunto fuzzy [Barros et al, 2010].

O módulo *defuzz*, da biblioteca *skfuzzy*, possui a função *skfuzzy.defuzz()*, que calcula o método de defuzzificação especificado. Para o vendedor 1 que vimos na seção anterior, sua saída *crisp* foi igual a 0.565, o que o classifica como "Churn A".

4.5 Resultados

Dada a base de dados que criamos, em 1000 vendedores, temos 816 foram classificados como "Churn A", 173 definidos como "Sem Churn" e os 11 restantes obtiveram a classificação de "Churn D". O valor de *churn* mais alto encontrado foi de 0.828, e o mais baixo igual a 0.213.

5 Conclusão

Os resultados obtidos podem ser considerados razoáveis, dado o conjunto de dados utilizado no modelo. No entanto, algumas limitações foram encontradas, principalmente em relação a base de regras. Por esta ser formulada manualmente, pode ser extenuante e custoso a formulação desta e, o trabalho aumenta conforme aumentam os parâmetros empregados. Além disso, *inputs* que são em formatos de datas são mais difíceis de serem modelados por funções de pertinência, já que é necessário serem convertidos para algum tipo numérico. Uma forma de contornar essas limitações seria de um sistema híbrido, formado por um sistema de inferência fuzzy, mas com uma base de regras parametrizada por outro tipo de método, como métodos de *machine learning* e outros algoritmos.

A ANEXO: Base de Regras

REGRA	QTD. ESTOQUE	DATA. ÚLTIMA VENDA	DATA. CADASTRO ESTOQUE	OUTPUT
1	ALTO	ALTO	ALTO	S/ CHURN
2	ALTO	MÉDIO	ALTO	S/ CHURN
3	ALTO	ALTO	MÉDIO	S/ CHURN
4	MÉDIO	ALTO	ALTO	S/ CHURN
5	MÉDIO	MÉDIO	MÉDIO	CHURN A
6	MÉDIO	ALTO	MÉDIO	S/ CHURN
7	ALTO	MÉDIO	MÉDIO	CHURN A
8	MÉDIO	MÉDIO	ALTO	CHURN A
9	ALTO	BAIXO	BAIXO	CHURN A
10	BAIXO	ALTO	BAIXO	CHURN A
11	BAIXO	BAIXO	ALTO	CHURN A
12	BAIXO	MÉDIO	MÉDIO	CHURN D
13	MÉDIO	BAIXO	MÉDIO	CHURN A
14	MÉDIO	MÉDIO	BAIXO	CHURN D
15	BAIXO	MÉDIO	ALTO	CHURN A
16	MÉDIO	BAIXO	ALTO	CHURN D
17	ALTO	BAIXO	MEDIO	CHURN A
18	BAIXO	ALTO	MÉDIO	CHURN A
19	MÉDIO	ALTO	BAIXO	CHURN D
20	ALTO	MÉDIO	BAIXO	CHURN A
21	BAIXO	BAIXO	BAIXO	CHURN D
22	ALTO	BAIXO	ALTO	CHURN A
23	BAIXO	MÉDIO	BAIXO	CHURN A
24	MÉDIO	BAIXO	BAIXO	CHURN A
25	BAIXO	ALTO	ALTO	CHURN A

Referências

[1] Bassanezi, R., Barros, L. (2010). Tópicos de Lógica Fuzzy e Biomatemática.

[2] Gafa, C. (2020). Fuzzy Inference System implementation in Python. https://towardsdatascience.com/fuzzy-inference-system-implementation-in-python-8af88d1f0a6e.

[3] Klir, G. J., Yuan, B. (1995). Fuzzy Sets and Fuzzy Logic: Theory and Applications. Pearson College Div.

[4] MathWorks. (n.d.). Mamdani and Sugeno Fuzzy Inference Systems. https://www.mathworks.com/help/fuzzy/types-of-fuzzy-inference-systems.html.

[5] Reichheld, F. F., W. Earl Sasser, J. (n.d.). Zero Defections: Quality Comes to Services. https://hbr.org/1990/09/zero-defections-quality-comes-to-services.

[6] Vafeiadis, T., Diamantaras, K. I., Sarigiannidis, G., Chatzisavvas, K. Ch. (2015). A comparison of machine learning techniques for customer churn prediction. Simulation Modelling Practice and Theory, 55, 1–9. https://doi.org/https://doi.org/10.1016/j.simpat.2015.03.003

[7] SalesForce. *How do you calculate revenue churn rate?*. Disponível em: https://www.salesforce.com/resources/articles/how-calculate-customer-churn-and-revenue-churn/.