Investigando o uso de RNN e GNN para previsão de churn de aplicativos

No Author Given

Universidade Federal de Viçosa - Campus Florestal, Minas Gerais, Brasil
{mateus.p.silva, gabriel.coimbra, fabricio.asilva}@ufv.br

Resumo. Empresas prestadoras de serviço perdem muito com o abandono de clientes, o que é chamado de *Churn* na literatura. Campanhas de fidelização oferecidas a todos os clientes podem ser muito custosas. Sendo assim, predizer quais clientes tendem a realizar *Churn* pode ser muito vantajoso. Neste trabalho, foi avaliado o desempenho de um modelo de rede neural de grafos (GNN) para predição de *Churn* em comparação com um modelo tradicional de floresta aleatória e um modelo profundo de rede neural recorrente (RNN) utilizando apenas dados não sensíveis de clientes em um contexto de abandono de serviço digital. Os resultados mostraram que o uso da rede neural de grafos mostrou-se benéfico: embora tenha tido a segunda melhor precisão (61% contra 63% da RNN) e segundo melhor *F1 Macro Score* (59.95% contra 61% da RNN), ela conseguiu melhor revocação da classe de *Churn* (68% contra 61.5% da *baseline* e 55% da RNN). Para trabalhos futuros, pretende-se estender a ideia do uso de redes neurais de grafos para problemas genéricos dados por listas de eventos utilizando-se ainda mais informações e recursos do modelo.

1. INTRODUÇÃO

Empresas prestadoras de serviço perdem muito com o abandono de clientes [Gupta et al. 2004]. Tal abandono é chamado de *Churn* na literatura. Além da estratégia de conseguir cada vez mais novos clientes, é importante evitar que os clientes atuais desistam do serviço. Uma estratégia comum é a utilização de programas de fidelização oferecidos a todos os clientes, beneficiando os consumidores por usarem o serviço por muito tempo. Isso, entretanto, é bastante custoso, pois as empresas acabam gastando recursos com clientes que já estão satisfeitos com o serviço prestado e não desejam realizar *Churn*.

Assim, uma estratégia que tem ganhado destaque é de antecipar quais são os clientes mais propensos a desistir do serviço, ou seja, predizer quais clientes provavelmente realizarão o *Churn*. Dessa forma, campanhas de fidelização poderão focar nesses clientes, e evitar gastos desnecessários com clientes que dificilmente fariam *Churn*. Além disso, é possível tentar prever o *Churn* em curto, médio ou longo prazo. A melhor escolha depende das necessidades da empresa. O tema de predição de *Churn* já foi bastante explorado na literatura em serviços de telefonia/Internet e cartões de crédito, que apresentam taxas de *Churn* altas, principalmente utilizando técnicas tradicionais de aprendizado de máquina supervisionado de classificação.

Nestas soluções, em geral dados sensíveis dos clientes são utilizados, como localização, informações financeiras, dados pessoais, uso detalhado dos serviços, dentre outros. Muitos países, inclusive o Brasil [Miragem 2019], avançam suas legislações e proíbem cada vez mais o uso desse tipo de dado. Além das questões legais quanto a obtenção desses dados, armazená-los também gera cada vez mais problemas jurídicos. Entretanto, não usar dados que sejam sensíveis aos clientes pode reduzir significativamente a qualidade da predição dos clientes que farão *Churn*, com o ganho de privacidade e proteção de dados. Assim, o uso de técnicas de aprendizado profundo se mostra bastante promissor, visto que permite um aproveitamento maior de informações não sensíveis, o que tende a compensar a falta de dados privados.

O objetivo deste trabalho é investigar se o uso de Redes Neurais de Grafos (GNNs) pode ser benéfico para a predição de *Churn* de serviços digitais sem uso de dados sensíveis. Para isso será avaliado um

modelo profundo de GNN. Serão utilizados apenas os dados de instalação e desinstalação de aplicativos móveis, ou seja, assume-se que o comportamento de instalação e desinstalação de outros aplicativos pode indicar propensão ao *Churn*. Por usar apenas tais dados, a solução aqui apresentada é bastante generalista quanto à natureza do serviço, bastando que ele aconteça através de um aplicativo móvel.

As soluções baseadas em aprendizado profundo foram comparadas com uma abordagem tradicional utilizando Florestas Aleatórias e uma abordagem de aprendizado profundo de rede neural recorrente (RNN), ambas comuns na literatura. Foram feitos testes considerando dados de 47.665 clientes de um banco digital. Os resultados mostraram que o modelo feitos utilizando redes neural de grafos apresenta a melhor revocação da classe de *Churn* (68%), embora tenha tido a segunda melhor precisão (61%) e também segundo melhor *F1 Macro Score* (59.95%). Tal resultado é promissor, pois geralmente a revocação é crítica em problemas de predição de .

Este texto está organizado da seguinte forma: a seção 2 descreve os trabalhos relacionados; a seção 3 descreve em mais detalhes os modelos e as tomadas de decisão; a seção 4 trata dos dados, avaliações e resultados obtidos e, por fim; a seção 5 discorre sobre as considerações finais.

2. TRABALHOS RELACIONADOS

O *Churn* pode ser entendido como o abandono de clientes de uma determinada empresa por qualquer motivo em um determinado período de tempo. A taxa de *Churn*, por sua vez, é a quantidade de clientes que abandonaram essa empresa dividida pelo total de clientes. Quanto menor a taxa de *Churn*, maior o gasto desnecessário em campanhas de fidelização generalistas. Entretanto, quanto maior a taxa de *Churn*, maior o prejuízo causado, e maior o desejo em diminuí-la.

Os trabalhos da literatura focam principalmente em problemas com taxas de *Churn* bastante altas. O notável mais comum contexto de taxa de *Churn* é o de telecomunicações (telefonia, Internet e televisão) [Jain et al. 2020] [Verhelst 2018], sendo trabalhos que normalmente apresentam taxas de *Churn* bastante altas, são possíveis de prever com modelos tradicionais e há muitos dados públicos disponíveis para o contexto. Outros contextos ainda incluem: aplicativos de redes sociais [Yang et al. 2018], aplicativos de *streaming* de música [Zhou et al. 2019], de cursos online massivos [Tan et al. 2018] e de empresas fornecedoras de cartão de crédito [Rajamohamed and Manokaran 2018].

O modelo de Floresta Aleatória foi exaustivamente utilizado em diferentes contextos de *Churn*, como o de telecomunicações [Ullah et al. 2019] [Idris et al. 2012]. Embora esses trabalhos tenham conseguido bons resultados, foram utilizados dados sensíveis dos clientes. Modelos de regressão logística e similares também foram utilizados nesse mesmo contexto [Jain et al. 2020], mas novamente com uso de informações sensíveis.

Quanto a modelos de aprendizado profundo, redes neurais recorrentes foram usadas [Hu et al. 2018] e apresentaram bons resultados, mas também com dados sensíveis. Embora redes neurais de grafos tenham sido usadas para diversos outros problemas [Jiang and Luo 2022] [Hamilton et al. 2017], ainda não há trabalhos com uso delas para predição de *Churn*. Também não foram encontrados trabalhos que abordam o problema de *Churn* para serviços digitais de forma generalista como o aqui apresentado. Neste presente trabalho, as técnicas de aprendizado profundo são utilizadas para prever *Churn* nesse contexto sem utilizar dados sensíveis dos clientes, apenas as informações de instalação e desinstalação de aplicativos.

3. SOLUÇÕES PARA PREDIÇÃO DE CHURN

Para as soluções, assume-se que estão disponíveis apenas informações sobre instalações e desinstalações de aplicativos móveis realizadas pelos clientes. Cada cliente é representado como uma sequência de eventos, sendo cada evento a instalação ou desinstalação de algum aplicativo. O evento contém um

identificador numérico único para cada aplicativo, a data de ocorrência e um indicador binário que difere se foi uma instalação ou desinstalação.

Com isso, definimos o conjunto de dados da seguinte forma:

Definição 3.1 *Conjunto de dados do evento.* Seja K_u o conjunto de 2-uplas (c, E), sendo c um indicador binário que é verdadeiro se o cliente fez *Churn* e falso caso o contrário e sendo E sequências de eventos e_i ordenados por tempo para o cliente u. Cada e_i é uma tupla (a, d, f) onde a especifica um número único de identificação para cada aplicativo móvel, d é um inteiro especificando o dia, começando em zero desde o início dos dados, que este evento aconteceu, e f é um indicador binário que é verdadeiro se o evento é uma instalação de aplicativo, e é falso se for uma desinstalação. Além disso, assume-se que h é a quantidade de números únicos de identificação para cada aplicativo móvel a.

3.1 Floresta aleatória (Baseline)

Para testar nossa hipótese de que o uso de mais informações não sensíveis pode melhorar o desempenho dos modelos, foi escolhido o modelo tradicional floresta aleatória. Por sua própria definição, tal modelo não trabalha com sequências de eventos, nem diretamente como as redes neurais recorrentes, nem mediante a transformações como as redes neurais de grafos.

Assim, a primeira etapa é transformar os dados para uma forma vetorial. Um simples pré-processamento é feito, de forma que apenas os aplicativos que estão presentes no dispositivo móvel no último dia de coleta dos dados são considerados. Os dados de entrada são, para cada cliente, um lista binária de tamanho N, sendo N o número de aplicativos únicos. Cada posição da lista é preenchida com verdadeiro, caso o respectivo aplicativo esteja instalado no dispositivo, e falso caso não esteja. Ou seja, pela própria natureza sequencial dos dados, muita informação é perdida no uso deste modelo. O alvo da predição do modelo é o indicador de *Churn* c de cada cliente.

3.2 Rede neural recorrente (RNN)

Em primeiro lugar, é importante explicar o formato da entrada RNN. As redes neurais recorrentes usam um tensor diferente das redes neurais *feed forward* (FNN) regulares. As FNNs trabalham com um tensor no formato $(nExemplos, nDimensao)$ em que $nExemplos$ representa o número de amostras de treinamento e $nDimensao$ representa os dados associados a cada amostra, que pode ter codificação binária ou mista. Para RNN, é necessária uma dimensão extra. Sendo assim, a forma do tensor é a seguinte: $(nExemplos, nPassosTemporais, nDimensao)$. Nesta configuração, $nPassosTemporais$ representa qual a localização na sequência em que os dados são armazenados em $nDimensao$. Além disso, como inspiração do uso de RNNs em processamento de linguagens naturais, podemos tratar a sequência de instalações e desinstalações como tendo uma ordem intrínseca, mesmo que nesse trabalho as instalações e desinstalações do mesmo dia sejam ordenados aleatoriamente como mostrado em 4.1.

Dado um conjunto de clientes K, conforme a Definição 3.1, é preciso aplicar filtros para descartar valores discrepantes em alguns aspectos. Como cada aplicativo terá codificação binária, o número de valores únicos de h, conforme a Definição 3.1, representará a dimensionalidade da entrada RNN posteriormente e será proporcional ao número de parâmetros e dados necessários para uma boa acurácia do modelo (i.e., sem sobre-ajustamento).

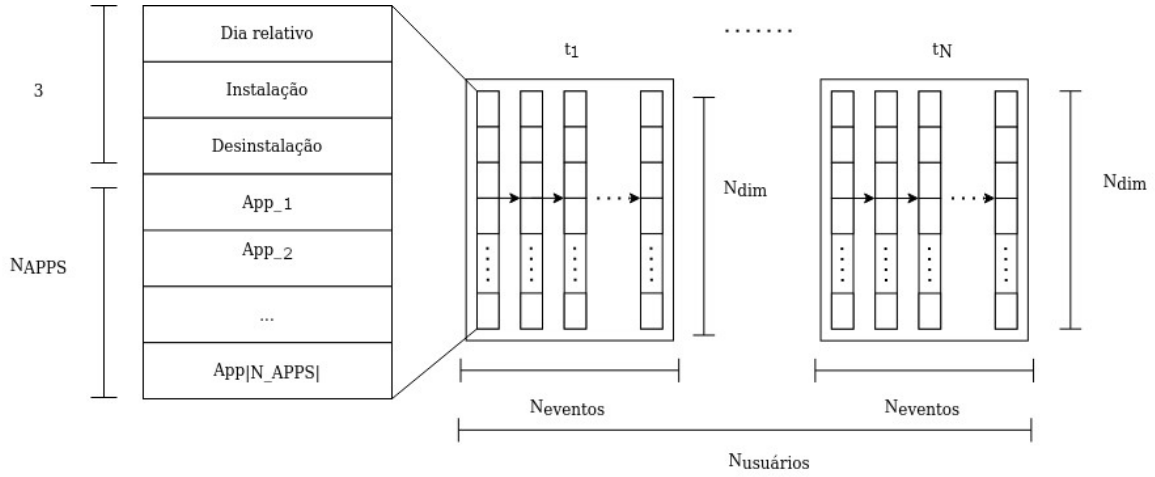

Fig. 1. Diagrama da representação dos dados utilizados na RNN.

Como pode ser visto na Figura 1, cada tensor t para um cliente foi definido da seguinte forma: t é um tensor bidimensional com forma $(N_{eventos}, N_{dimensao})$. Onde $N_{eventos}$ é máximo possível de eventos para todos os usuários e $N_{dimensao}$ é igual ao número de aplicativos únicos $N_{apps} + 3$, sendo que três posições são utilizadas para armazenar metadados dos eventos. Como nem todo cliente tem a mesma quantidade de instalações/desinstalações, os tensores sobressalentes são preenchido com zeros. Com isso é possível extrair um vetor na posição $t_{m,n}$, onde $t_{m,n} \in \mathbb{R}^{N_{apps}+3}$. Na primeira posição desse vetor, $t_{m,n,0}$ é usada para indicar em que dia relativo esse evento ocorreu da seguinte forma: sendo $maximo(d)$ o maior valor de d da definição 3.1, a posição $t_{m,n,0}$ terá valor $d_j/maximo(d)$ onde $j \leq N_{eventos}$. A segunda e terceira posição $t_{m,n,1}$ e $t_{m,n,2}$ são usadas para codificar instalação e desinstalação (note que não seria possível utilizar apenas uma posição binária para isso, visto que pela própria definição de redes neurais, os valores 0 são desconsiderados no aprendizado. Assim, ou a desinstalação seria ignorada, ou a instalação). Caso o evento seja uma instalação $t_{m,n,1} = 0$ e $t_{m,n,2} = 1$, caso contrário, os valores são invertidos. As posições restantes t_m, n, x onde $3 \leq x \leq N_{apps}$ é uma informação codificada *One-Hot* de qual aplicativo instalado ou desinstalado foi considerado naquele tensor, ou seja, apenas um aplicativo é permitido por tensor. Por exemplo sendo o n-ésimo evento do cliente m, $T[m, n, :] = [2/40, 1, 0, 0, 0, 0, 1]$ indica que o aplicativo de id = 3 ($T[m, n, 3 + 3]$) foi desinstalado no dia 2 pelo cliente m e que o maior número distinto de dias para todos clientes é 40.

3.3 Rede neural de grafos (GNN)

O primeiro passo para treinar uma rede neural de grafos (GNN) é a criação dos grafos. A criação dos grafos ocorre de forma independente entre os clientes, sendo que para cada cliente, um grafo direcionado é gerado. Cada vértice v representa um aplicativo, e cada aresta direcionada que conecta v_m a v_n representa a influência que a instalação de v_m gera sobre a possibilidade de instalação de v_n.

Com as tuplas mapeadas, é possível criar os grafos. O grafo gerado tem matriz de adjacência $A_{N\times N}$. A matriz é iniciada com todas as arestas com o peso 0, ou seja, desconectadas, como mostrado na Figura 2. Tal exemplo será seguido doravante.

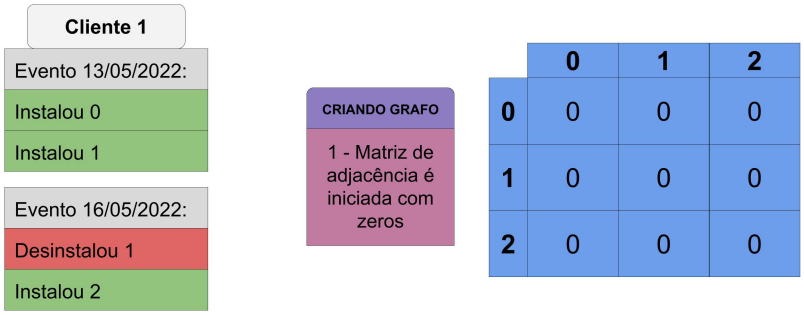

Fig. 2. 10.

Todas as tuplas são agrupadas em grupos g_d de acordo com a data d, ou seja, são criados grupos de instalações e desinstalações que ocorreram no mesmo dia. Os valores da matriz de adjacência são então incrementados conectando todas as tuplas de um mesmo grupo, pois assume-se que os aplicativos instalados e desinstalados no mesmo dia tem forte correlação. Assim, o peso das arestas correspondentes a esses aplicativos são incrementados utilizando a função:

$$a_{k,l} += P(k) \times P(l)$$

onde k e l são todas as tuplas diferentes de um grupo g_d, ou seja, são instalações e desinstalações de aplicativos que aconteceram no mesmo dia. A função P define o peso do tupla, que é -1 se for uma desinstalação (ou seja, se o binário f for falso), e 1 se for uma instalação de aplicativo (se o binário f for verdadeiro). Dessa forma, é esperado que dois aplicativos que foram desinstalados ou instalados tenham uma correlação positiva, e que um aplicativo instalado e outro desinstalado tenham uma correlação negativa.

Seguindo o exemplo da Figura anterior, as instalações dos aplicativos 0 e 1 do dia 13/05 são conectadas. O peso das arestas $0 \rightarrow 1$ e $1 \rightarrow 0$ são somadas com o valor de 1, visto que são duas instalações, ou seja, $P(0) = 1$ e $P(1) = 1$, como mostrado na Figura 3:

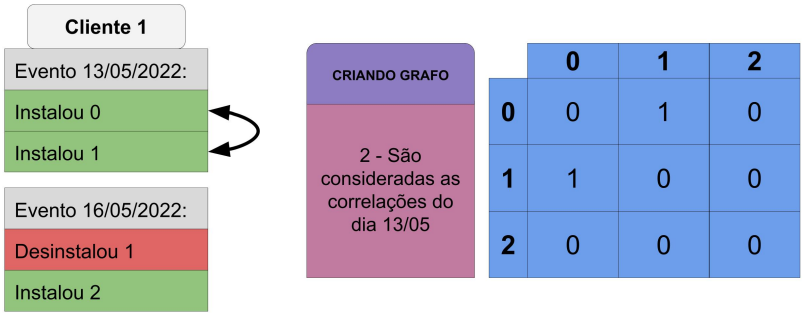

Fig. 3. As posições 0,1 e 1,0 da matiz são acrescidas com 1.

O próximo passo é considerar o dia 16/05. A desinstalação do aplicativo 1 é considerada em conjunto com a instalação do aplicativo 2. Note que o peso acrescido será de -1 nas arestas $1 \rightarrow 2$ e $2 \rightarrow 1$, visto que são uma instalação e uma desinstalação, ou seja, $P(1) = -1$ e $P(2) = 1$, como mostrado na Figura 4:

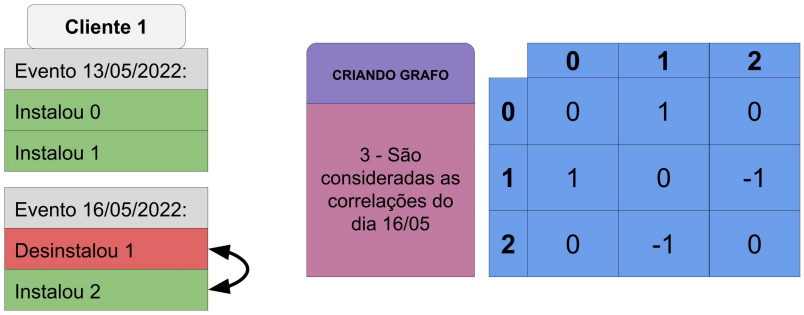

Fig. 4. As posições 1,2 e 2,1 da matiz são acrescidas com -1.

Ou seja, depois de conectadas as instalações e desinstalações de mesmo dia, o grafo do exemplo ficará como mostrado na Figura 5.

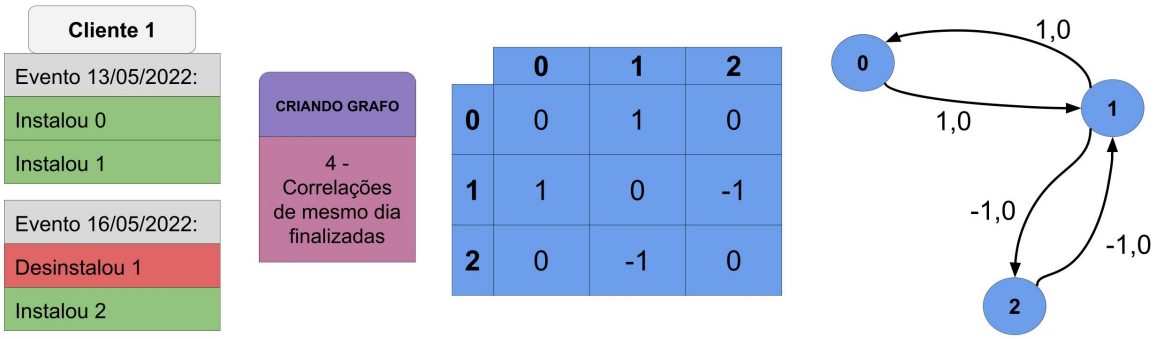

Fig. 5. Grafo com instalações e desinstalações de mesmo dia consideradas.

Depois disso, as tuplas de datas subsequentes são consideradas. Os grupos de tuplas são ordenados de forma crescente a data d, e cada grupo de tuplas g_d é considerado em par com seu próximo $g_d + 1$. Então, a matriz de adjacência tem seus pesos atualizados segundo a função:

$$a_{m,n} += \frac{P(m) \times P(n)}{D(m) - D(n) + 1}$$

Sendo m todas as tuplas do primeiro grupo considerado (g_d), n todas as tuplas do grupo subsequente em data ($g_d + 1$) e D uma função que retorna o valor de d da tupla dada. Assim, as correlações entre instalações e desinstalações que ocorrem em datas subsequentes no conjunto de dados são consideradas utilizando uma ideia parecida com a anterior, porém com o acréscimo de informação da distância temporal entre as datas - ou seja, é dado um peso maior para eventos próximos.

Seguindo o exemplo dado, é considerada agora a instalação do aplicativo 0 no dia 13/05 com a desinstalação do aplicativo 1 no dia 16/05. O divisor da fórmula citada será $16 - 13 + 1 = 4$, e o dividendo de -1, pois $P(0) = 1$ e $P(1) = -1$. Logo, o peso da aresta $0 \rightarrow 1$ será acrescido em -0,25, como é mostrado na Figura 6. Note que apenas a aresta $0 \rightarrow 1$ é acrescida, e não a $1 \rightarrow 0$

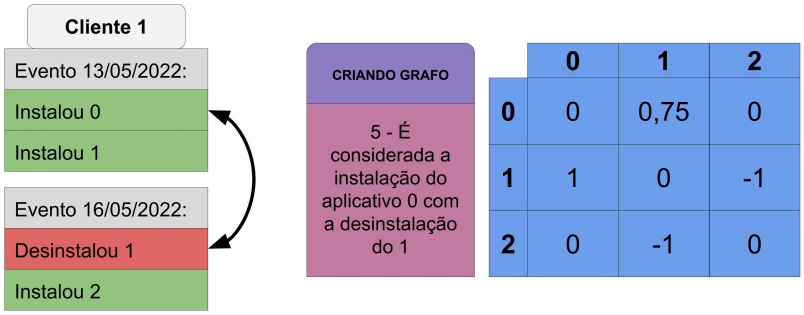

Fig. 6. As posição 0,1 da matriz foi acrescida com -0,25.

Continuando o exemplo, agora é considerada agora a instalação do aplicativo 0 no dia 13/05 com a instalação do aplicativo 2 no dia 16/05. O divisor da fórmula citada será $16 - 13 + 1 = 4$, e o dividendo de 1, pois $P(0) = 1$ e $P(2) = 1$. Logo, o peso da aresta $0 \rightarrow 2$ será acrescido em 0,25, como é mostrado na Figura 7.

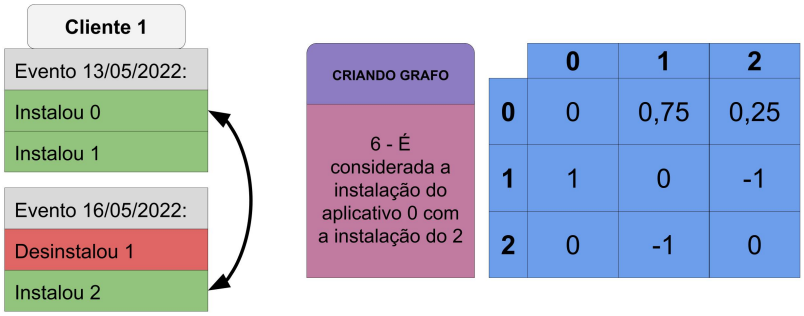

Fig. 7. As posição 0,1 da matriz foi acrescida com -0,25.

Realizando a mesma tarefa para a aresta $1 \rightarrow 2$, o grafo resultante será o da Figura 8.

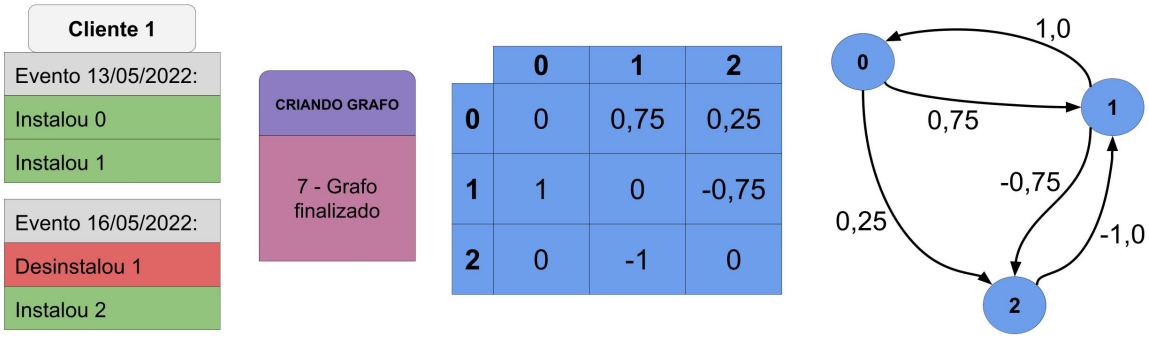

Fig. 8. As posição 0,1 da matriz foi acrescida com -0,25.

Com todos os grafos criados, já é possível treinar o modelo. O modelo utilizado consiste em três camadas: uma densa de ativação *relu* de N unidades, uma de *pooling* de soma e outra densa de ativação

sigmoid de 1 unidade. A primeira camada recebe diretamente a matriz de adjacência dos grafos. A camada densa de ativação recebe diretamente a matriz de adjacência do grafo, e utiliza a função de ativação *relu* por ser conhecida por funcionar bem para camadas internas. A segunda camada realiza *pooling* de soma, útil para somar o impacto de cada aplicativo no *Churn*. Ao fim, a terceira camada utiliza a função de ativação *sigmoid* para retornar um valor entre 0 e 1, a fim de realizar a classificação necessária de *Churn*.

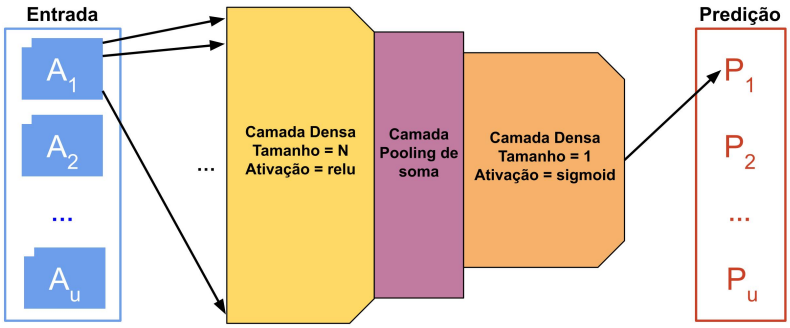

Fig. 9. Modelo de rede neural de grafos.

4. AVALIAÇÃO

4.1 Os dados

Para avaliar as soluções, foram utilizados dados reais de usuários de um banco digital no Brasil. Os dados foram obtidos por meio de um acordo de confidencialidade com uma empresa parceira. A coleta ocorreu por meio de um software agente instalado em conjunto com o aplicativo de um banco digital no *smartphone* do cliente. Portanto, a coleta dos dados usados nesse trabalho começa quando o cliente instala o aplicativo e passa a utilizá-lo. O software agente captura o momento de abertura do aplicativo do banco digital e também a lista de aplicativos instalados no *smartphone*. Com isso é possível gerar uma lista de eventos sobre aplicativos (i.e., aplicativos foram instalados ou desinstalados) para cada cliente e saber qual a última data que o cliente abriu o aplicativo do banco digital. Os dados representam 47.665 clientes, coletados do período de 01-01-2021 a 30-07-2021. Além disso, foi possível coletar o último dia que o usuário acessou o aplicativo do banco digital em um período posterior, no dia 31-01-2022. É importante destacar que, devido a necessidade de economia de bateria, o evento só envia a lista de aplicativos instalados uma vez por dia. Nesse caso, se mais de um aplicativo foi instalado ou desinstalado em um mesmo dia, não se sabe a ordem que esses eventos aconteceram.

Foram considerados *Churners* clientes que ficaram 5 meses sem abrir o aplicativo, pois é desejável ter alta confiança na decisão de quais clientes serão classificados como *Churners*. Esse intervalo de tempo maior foi utilizado pois existe a possibilidade de que o banco digital estudado não seja o preferido do cliente, ou seja utilizado somente como conta de investimentos por exemplo, e nesse caso o acesso esporádico pode ser possível. Com isso, foram utilizados dados que podem anunciar uma tendência de *Churn* no médio prazo (i.e., 5 meses). Isso pode ser interessante em corporações pois pode ser necessário um tempo com engajamento de uma empresa para evitar o *Churn* no futuro. Com esta definição de *Churn*, foram registrados 11.504 clientes *Churners*.

Para evitar sobre-ajustamento e acelerar o tempo de treinamento dos algoritmos, foram mantidos apenas os 95% aplicativos com maior número de instalações e desinstalações. Esse filtro manteve os 1.058 aplicativos mais frequentes do conjunto de dados.

Além disso, os identificadores a das tuplas foram mapeados para números naturais de 1 a h (sendo h a quantidade de identificadores a únicos, veja a Definição 3.1) como mostrado na Figura 10. Tal mapeamento é feito a fim de reduzir o tamanho necessário da matriz de adjacência dos grafos da rede neural de grafos (veja a Seção 3.3) e do tensor $t_{m,n}$ da rede neural recorrente (veja a Seção 3.3), cujos tamanhos são proporcionais ao maior identificador a a fim de reduzir tempo de execução. A escolha do mapeamento é feita por ordem de aparição no conjunto de dados, já que é a mais simples e qualquer ordem teria o mesmo resultado.

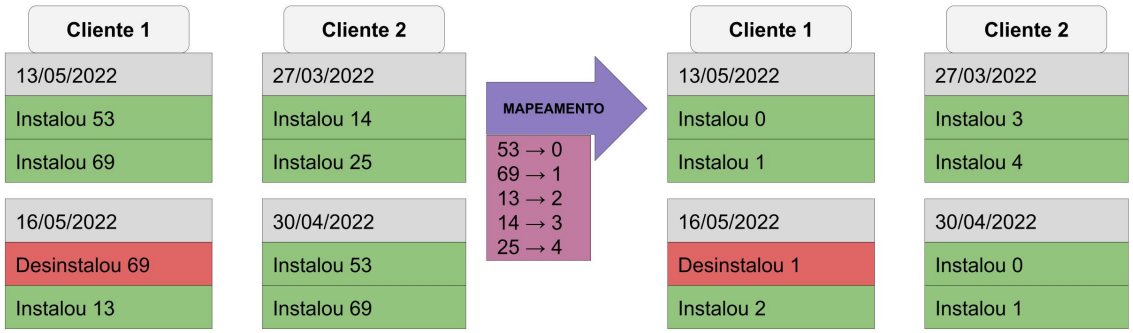

Fig. 10. Exemplo de mapeamento de identificadores de aplicativos a.

Além isso, filtramos também os clientes que não tenham pelo menos dois meses de dados gravados, pois a quantidade de eventos deles poderia ser insuficiente para o treinamento.

Quanto a limpeza dos dados específicos para a RNN, foram filtrados clientes com pelo menos 5 a 40 dias únicos. Como clientes com poucos dias distintos pode fazer com que o modelo não tenha dados suficientes, uma vez que não temos conhecimento da sequência de instalações/desinstalações de mesmo dia. Além disso, clientes com muitos dias distintos também podem confundir a rede, pois o número de células recorrentes precisará corresponder ao número de eventos para cada cliente. Portanto, filtramos clientes em que o número de eventos ($|N_{eventos}|$) esteja dentro do quantil 10% (55 eventos) e 95% (700 eventos).

4.2 Resultados

Como já é bastante conhecido na literatura, modelos de aprendizado de máquina não trabalham bem com conjuntos de dados desbalanceados. Os dados foram então balanceados utilizando-se o método de subamostragem. Inicialmente eram 47.665 clientes sendo 11.504 *Churners*, ou seja, apenas 24%. Depois do balanceamento restaram 23.008 clientes, sendo metade de cada classe.

Todos os modelos e testes foram feitos utilizando-se ambiente de desenvolvimento Python. Quanto a floresta aleatória, foi utilizada a biblioteca SKLearn. Para o modelo de rede neural recorrente, foi utilizado o popular Tensorflow. Por fim, a rede neural de grafos foi construída utilizando-se a biblioteca Spektral, que permite criação de redes neurais de grafos sobre o já citado Tensorflow. A floresta aleatória levou cerca de 1 minuto para treinamento, e a rede neural recorrente levou 15 minutos de treinamento por época, ambos via Google Colab. Quanto a rede neural de grafos, foi utilizado processamento em GPU via Docker. Na máquina testada (Nvidia Geforce GTX 1050 TI), cada época da rede levou 30 minutos em média para o treinamento.

Quanto aos parâmetros dos modelos, a Floresta Aleatória utilizou os parâmetros padrão da biblioteca sklearn [1]; a rede neural recorrente (veja a seção 3.2) utilizou taxa de aprendizado de 0,001, 4

[1] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

épocas e tamanho do *batch* como 32; e a rede neural de grafos (veja a seção 3.3) foi inicializada com pesos aleatórios e utiliza *EarlyStopping* com 10 de paciência, taxa de aprendizado de 0,001, tamanho do *batch* de 32 e a quantidade de épocas como 10.

Todos os modelos foram testados com validação cruzada com 10 divisões, resultando em treinamentos com 90% dos dados e testes com 10%. Tais divisões não são balanceadas, visto que devido a escolha aleatória de dados balanceados gera naturalmente divisões razoavelmente balanceadas. Os resultados nos gráficos mostram o valor médio e o intervalo de confiança de 95%. Quanto as redes neurais, as melhores épocas foram consideradas.

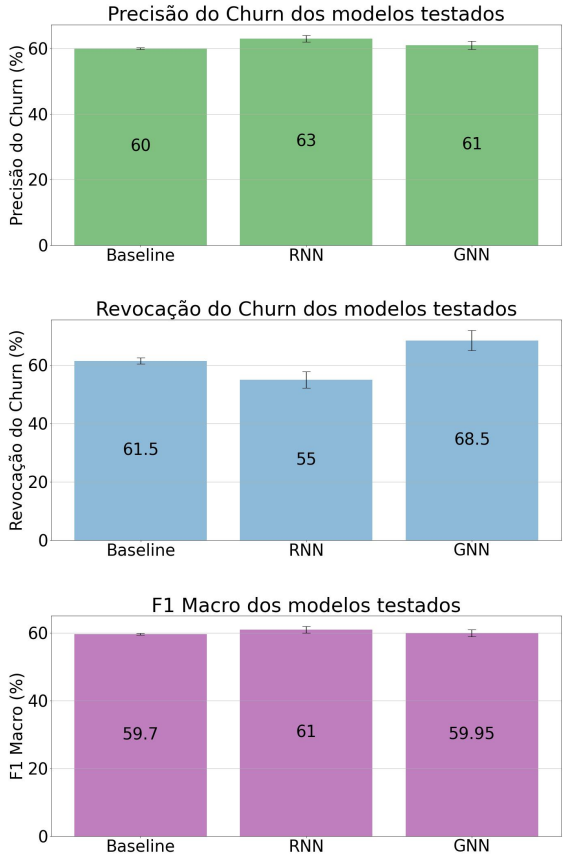

Fig. 11. Resultados obtidos utilizando os três modelos.

Dos três modelos, o que melhor se destaca tanto em precisão quanto em F1 Score é a rede neural recorrente (Veja a seção 3.2), com precisão de 63% e F1 Score de 61%. Entretanto, a rede neural de grafos destacou-se bastante em revocação, conseguindo 68%. Isso significa que, do conjunto de todos os *Churners*, o modelo foi capaz de classificar corretamente 68% deles. No contexto de predição de *Churn*, tal métrica pode ser ainda mais importante que a precisão, visto que o objetivo é classificar corretamente o maior número de *Churners* possível sem sacrificar muito a precisão.

5. CONSIDERAÇÕES FINAIS

Foi observado que o modelo de rede neural de grafos parece bastante promissor para a predição de *Churn* em ambientes que não permitem uso de dados sensíveis dos clientes. Isso se deve ao fato de

que, embora as redes neurais recorrentes apresentem maior precisão e F1 Score, as redes neurais para grafos apresentam melhor revocação na predição de *Churn*.

Além disso, é importante observar que os modelos de redes neurais de grafos podem ser usados para codificação de ainda mais informações dos clientes na forma de grafos - sejam dados sensíveis ou não - na forma de atributos de nó e arestas ou ainda como dados estáticos recebidos ao lado do grafo no treinamento. Tal abordagem, com uso de mais dados, pode melhorar ainda mais o desempenho dos modelos.

Para trabalhos futuros, pretende-se estender a ideia do uso de redes neurais de grafos para problemas genéricos dados por listas de eventos, seja de classificação (como é o caso do problema de predição de *Churn*) quanto de regressão. Tal extensão poderia fazer uso de diferentes abordagens de criação automática de grafos, como uso de programação genética para criação das funções de peso das matrizes de adjacência e uso de algum modelo simples de rede neural para *fitness*. Além disso, seria interessante a análise de algumas características de grafos (como *Clique*) para visualização e entendimento humano dos grafos para análise de negócios.

REFERÊNCIAS

Gupta, S., Lehmann, D. R., and Stuart, J. A. Valuing customers. *Journal of marketing research* 41 (1): 7–18, 2004.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. *Advances in neural information processing systems* vol. 30, 2017.

Hu, J., Zhuang, Y., Yang, J., Lei, L., Huang, M., Zhu, R., and Dong, S. prnn: A recurrent neural network based approach for customer churn prediction in telecommunication sector. In *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, pp. 4081–4085, 2018.

Idris, A., Rizwan, M., and Khan, A. Churn prediction in telecom using random forest and pso based data balancing in combination with various feature selection strategies. *Computers & Electrical Engineering* 38 (6): 1808–1819, 2012.

Jain, H., Khunteta, A., and Srivastava, S. Churn prediction in telecommunication using logistic regression and logit boost. *Procedia Computer Science* vol. 167, pp. 101–112, 2020.

Jiang, W. and Luo, J. Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications*, 2022.

Miragem, B. A lei geral de proteção de dados (lei 13.709/2018) e o direito do consumidor. *Revista dos Tribunais* vol. 1009, 2019.

Rajamohamed, R. and Manokaran, J. Improved credit card churn prediction based on rough clustering and supervised learning techniques. *Cluster Computing* 21 (1): 65–77, 2018.

Tan, F., Wei, Z., He, J., Wu, X., Peng, B., Liu, H., and Yan, Z. A blended deep learning approach for predicting user intended actions. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, pp. 487–496, 2018.

Ullah, I., Raza, B., Malik, A. K., Imran, M., Islam, S. U., and Kim, S. W. A churn prediction model using random forest: analysis of machine learning techniques for churn prediction and factor identification in telecom sector. *IEEE access* vol. 7, pp. 60134–60149, 2019.

Verhelst, T. Churn prediction and causal analysis on telecom customer data, 2018.

Yang, C., Shi, X., Jie, L., and Han, J. I know you'll be back: Interpretable new user clustering and churn prediction on a mobile social application. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. pp. 914–922, 2018.

Zhou, J., Yan, J.-f., Yang, L., Wang, M., and Xia, P. Customer churn prediction model based on lstm and cnn in music streaming. *DEStech Transactions on Engineering and Technology Research* vol. 5, 2019.