FUNDAÇÃO GETULIO VARGAS
ESCOLA DE ADMINISTRACAO DE EMPRESAS DE SÃO PAULO

Programa Institucional de Bolsas de Iniciação Cientifica (PIBIC)

Métodos Quantitativos Aplicados ao Mercado Financeiro

ENRICO TORRIERO
JOÃO LUIZ CHELA

São Paulo – SP
2020

Índice:

Métodos Quantitativos Aplicados ao Mercado Financeiro

1 RESUMO

Grandes inovações cientificas se deram pela implementação de *Machine Learning* de forma efetiva e incisiva. Temos casos marcantes, como quando o computador da empresa norte americana IBM, o *"Deep Blue",* ganhou do então atual campeão mundial de xadrez, Garry Kasparov, em uma partida de seis jogos. Sendo que, anos após a partida, Kasparov admitiu que desde aquela época seria "impossível, para humanos, competir" (Kasparov, 2017).

Existem, no entanto, outras aplicações, que estão por traz do nosso dia-a-dia, como o reconhecimento de células possivelmente cancerígenas, que ajudam oncologistas do mundo todo no diagnostico de tumores malignos (Fakoor, 2013).

Sendo assim, a aplicação de algoritmos de *Machine Learning,* mais especificamente redes neurais, poderiam contribuir de forma positiva na precisão dos modelos de *Credit Scoring*, nos quais são geralmente atualizadas regressões logísticas para realizar a inferência do que seria um "bom" ou "mau" pagador (Sicsú, 2010). Acreditando que, o poder computacional proporcionado por este tipo de modelo matemático poderia gerar uma capacidade preditiva relevantemente maior do que os métodos mais convencionais de previsão de risco de crédito.

Com isso, foram desenvolvidos dois modelos de Credit Scoring utilizando o mesmo banco de dados para conseguir distinguir qual dos dois geraria um resultado melhor. Utilizando, portanto, um modelo com redes neurais, e outro com regressão logística. Ambos foram aplicados utilizando a linguagem de programação R.

Chegando na conclusão que, mesmo que o poder preditivo atrelado a algoritmos de *Machine Learning*, e consequentemente, redes neurais, possa ser maior do que métodos estatísticos como a regressão logística, muitas vezes, essa potencialização dos resultados pode virar apenas um gasto computacional não justificado, principalmente para aplicações que não são tão demandantes, como é o caso de Credit Scoring.

Palavras-Chave

Credit Scoring, Redes Neurais, Estatística, Modelagem Preditiva

2 INTRODUÇÃO

O ganhador do prêmio Nobel de física em 1922, Niels Bohr, disse, “É difícil fazer previsões, especialmente sobre o futuro” (Bohr, 1922). E esse é o desafio que empresas do mundo todo enfrentam de diferentes formas.

Sendo assim, conforme vão surgindo novas tecnologias e computadores com um maior poder de processamento. Tarefas que antes eram impossíveis de ser realizadas na prática se tornam viáveis (Foote, 2017). Além das aplicações na medicina, como já citado acima, empresas também conseguem extrair grandes benefícios de modelagem preditiva.

> “Exemplos [de inteligência artificial] podem ser encontrados em todo lugar: As máquinas globais da Google usam AI para interpretar buscas humanas. Empresas de cartão de crédito usam para monitorar fraude. Netflix usa para recomendar filmes para os assinantes. E o mercado financeiro usa para suportar biliões de trocas” (Kuhn, 2013, tradução própria)

Atualmente, grandes instituições de ensino, como o MIT e Stanford, estão oferecendo em suas plataformas virtuais os vídeos de suas aulas, assim como o material utilizado, de forma gratuita, procurando incentivar pesquisas e novas descobertas em relação ao tema. Existem, também, empresas multinacionais como o Google e a Amazon que oferecem poder computacional em seus serviços de *Cloud* de forma gratuita, sendo suficiente para pelo menos começar no desenvolvimento de redes neurais, e ate utilizar de forma relevante. Editoras como a Springer tem series de livros dedicadas exclusivamente a modelagem preditiva e também ao desenvolvimento de algoritmos de inteligência artificial. Além da grande comunidade que esta presente em fóruns de discussão específicos ao tema, como GitHub e Stackoverflow, compartilhando repositórios com os códigos criados e também tirando possíveis dúvidas que venham a surgir.

Competições a respeito do tema vem sendo cada vez mais difundidas, como no site Kaggle, que além de distribuir bancos de dados para estudo e aperfeiçoamento de modelos, também oferece premiações em dinheiro, junto com empresas parceiras, para os times que apresentarem modelos mais eficientes em tarefas escolhidas pelas empresas. Outro exemplo, grandes empresas de consultoria como a BCG também fomentam competições que dão aos ganhadores vagas de estágio

e prêmios que são usados para o desenvolvimento pessoal dos vencedores dentro do tema de *machine learning*.

Com isso, este trabalho tem por objetivo desenvolver um modelo de *Credit Scoring*, que consiste em um algoritmo de classificação de clientes de um banco de varejo em bons e maus pagadores, utilizando redes neurais e comparando com o método mais utilizado no mercado que é a regressão logística (Sicsu, 2020). Os modelos foram programados utilizando a linguagem R e o banco de dados utilizado veio do Kaggle, site já citado acima que oferece dados de forma gratuita para desenvolvimento de modelos de inteligência artificial e modelagem estatística. Os dados em questão são de um banco de varejo global, com aproximadamente 120 variáveis e 300 mil observações.

Para avaliar a eficácia dos modelos serão utilizados dois testes, AUROC e Spielgelhalter, para testar, respectivamente, o quão bem o modelo divide a amostra entre bons e maus pagadores e quanto o risco de ser um mau pagador (*default risk*) reflete a realidade.

3 TEORIA

A ideia de uma rede neural, um modelo matemático que tentaria imitar a logica de funcionamento do cérebro humano, vem de muitos anos atrás. Os primeiros a desenvolver um programa arcaico foram Walter Pitts e Warren McCulloch. Eles desenvolveram a ideia de *"threshold logic"* (Foote, 2017), que consiste na ativação de um "neurônio" (ou nó) a partir de um valor pré-determinado, ou seja, essa fonte de processamento receberia uma soma de valores (*inputs*), e dependendo de quanto for o resultado, ele atribuirá um certo valor a ele (*output*).

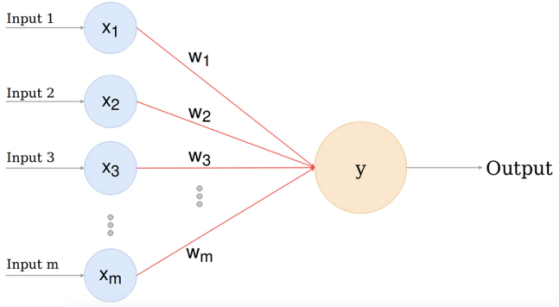

Figura 1 - Arunava

No exemplo que temos na figura, são apresentadas variáveis, x_1, x_2, x_3, até x_m. Dependendo de quanto for a soma dos valores ligados a cada uma e multiplicados pelos respectivos pesos "w",

teremos o *output* de 0 (falso) ou 1 (verdadeiro). Essa logica de ter um valor pré-determinado que desencadearia novos processos é o principal conceito dentro de uma rede neural.

$$y = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \dots \\ x_m \end{bmatrix} . [w_1\ w_2\ w_3\ \dots\ w_m]$$

Esse valor, "y", então seria colocado em uma função de ativação e então calculado o output do modelo. Existem diversas funções de ativação para processos diferentes (Aggarwal, 2018). Atualmente, e também usada neste projeto, uma das funções de ativação mais comum é a função tangente hiperbólica, ou apenas Tahn (Sicsu, 2018).

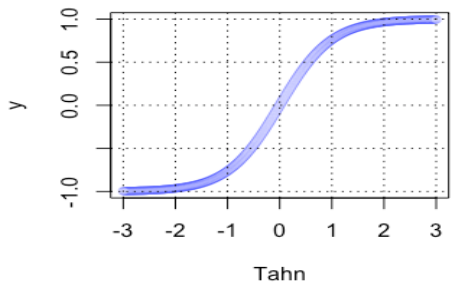

Figura 2 - Produzida pelo autor usando R

A partir da ideia do *perceptron*, que seria a arquitetura mais rudimentar de uma rede neural, com apenas um "nó" de processamento. Desenvolvedores começaram a adicionar mais camadas de processamento, chegando em redes mais complexas como a da foto abaixo:

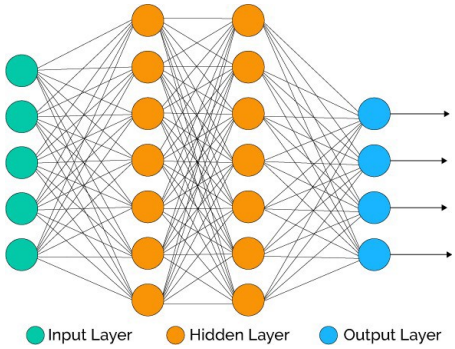

Figura 3 – McDonald, 2017

Com arquiteturas mais complexas e fortalecendo o campo. Em 1960, Henry Kelley trouxe a ideia de *back propagation* (Foote, 2017), que seria uma forma de, após as informações terem passado pela rede neural, e um resultado ser encontrado, seria possível comparar o resultado com o esperado, trazendo a ideia de erro, e treinamento para um modelo.

$$\text{Erro} = \sum_1^n [s(x_i) - y(x_i)]^2$$

$s(x_i)$ – output da rede neural

$y(x_i)$ – valor esperado

Portanto, quando menor a diferença entre o output e o valor esperado, mais bem treinado está o modelo. Caso não esteja suficientemente ajustado o modelo, é utilizado uma serie de derivadas parciais para atualizar os pesos, e assim, minimizar o Erro encontrado.

Porém, mesmo teorizando o conceito, não tinha sido possível a sua implementação. A primeira vez que foi possível de fato utilizar o *back propagation* de forma eficaz foi em 1989. Quando Yann LeCun fez um modelo que conseguia diferenciar números escritos a mão no papel (Foote, 2017). Para conseguir tal feito, ele utilizou uma técnica chamada de *convolution neural networks*, criada por Kunihiko Fukushima. Esse tipo de modelo separa imagens grandes em amostras menores, e assim decifra o que está contido nelas.

Em torno de 1999, a potência e o preço de placas de vídeo, responsáveis por decodificar informação para imagens, tiveram uma significativa melhora (Foote, 2017). Facilitando a acessibilidade das redes neurais e sua força de previsão. No entanto, com todo o progresso computacional, surgiu o *"Vanishing Gradient Problem"*, que era causado pelo treinamento exagerado de modelos, que ocorre quando o algoritmo encontra um ponto de mínimo local, e não necessariamente o ponto ótimo. Ou, que extrapole suficientemente e tenda ao infinito. Com isso, era necessário estabelecer um ponto para o computador parar de passar informação pela rede neural. Nos dias de hoje, é necessário ponderar o quanto se deve treinar um modelo, se separar pouco tempo para ele se adaptar, não é possível obter os resultados requisitados. Nem se deixarmos por tempo indeterminado, pois o resultado seria extrapolado.

Nos últimos tempos, a coleta e armazenamento de dados vem ficando cada vez mais simples e barata, dado que as companhias, governos, redes sociais, pessoas no geral, estão gerando mais

dados, das formas mais variadas a cada ano que passa (Thompson, 2019). E, como redes neurais demandam uma quantidade grande de dados para conseguir fazer previsões de forma eficaz, o avanço nos estudos do chamado *"Big Data"* tem um impacto muito positivo nesse campo de estudos.

> "Big Data – grandes conjuntos de dados que são produzidos por pessoas usando a internet, e que só pode ser armazenado, entendido, e utilizado com a ajuda de métodos e artifícios especiais" (*Cambridge Dictionary*, tradução propria)

Outro ponto que também vem contribuindo de forma positiva para a modelagem de redes neurais é a computação em nuvem, que possibilita qualquer um, desde que tenha uma conexão com a internet, de criar modelos e treina-los em servidores externos. Garantindo que o campo de estudo evolua e mais pessoas se adequem às novas tecnologias, principalmente em países como o Brasil, onde computadores com um grande poder de processamento tem um preço muito alto.

No decorrer do tempo é possível observar certas aplicações do *deep learning* que marcaram o campo de estudos. Em 2012, nos laboratórios "X" da Google, foi criado um modelo que utilizava *unsupervised learning* (aprendizado não supervisionado). Em modelos de redes neurais convencionais as imagens, ou o que estiver sendo utilizado como dado, são rotulados, então o modelo passa por um processo de treinamento com o que foi fornecido até que chegue no resultado almejado. Porém, no aprendizado não supervisionado os dados não são rotulados, cabendo ao próprio modelo identificar padrões que segmente as informações (Markoff, 2012). Com isso, a Google fez um algoritmo que pode parecer sem necessidade, porem teve uma grande representatividade de demonstrar o que era possível realizar com redes neurais. O modelo deles, que tinha a disposição 16.000 processadores de computador, era usado para reconhecer imagens de gatos na internet. Sendo o primeiro a realizar um feito como esse nessas proporções.

4 MÉTODOS

4.1 Banco de Dados

Como já dito anteriormente, redes neurais se beneficiam de grandes quantidades de dados para seu treinamento. Sendo assim, seria necessário a obtenção de uma quantidade significante de observações. Além disso, outro problema seria da própria natureza dos dados necessários para a criação de um modelo de Credit Scoring. Geralmente, são dados sensíveis, que as pessoas não abririam facilmente, por se tratar de variáveis como renda, endereço, protestos etc.

Com isso, fui atrás de plataformas destinadas ao estudo de modelagem preditiva e *machine learning* no geral. Achando, então, algo relevante no site Kaggle. Lá, uma instituição financeira tinha patrocinado uma competição em 2018 que procurava achar o grupo que desenvolvesse o melhor modelo de previsão de risco de crédito. Sendo assim, disponibilizou de forma gratuita, um banco de dados que, bruto, continha mais de 120 variáveis e 300 mil observações. Aparentemente mais do que o suficiente para o treinamento de uma rede neural. No entanto, existem dois problemas aparentes nesse banco de dados. O primeiro é que a empresa não disponibilizou o significado de cada variável, existem algumas que apenas com o nome é possível inferir sua necessidade, porem outras não estão escritas de forma explicita. O que fez com que a análise qualitativa das variáveis não fosse 100% coerente. Fazendo com que sejam utilizadas técnicas quantitativas de forma excessiva. O segundo problema, é a proporção observada de bons e maus pagadores na variável dependente, sendo aproximadamente 92% das observações bons pagadores e apenas 8% maus pagadores. Isso dificulta o treinamento de modelos preditivos e é especialmente desfavorável ao se tratar de redes neurais (Sicsu, 2020). Isso prejudica diretamente os resultados encontrados utilizando o teste de AUROC, que mede o quão eficiente o modelo é em dividir a amostra nas categorias pretendidas, no caso, bons e maus pagadores.

Porém, o banco de dados é fornecido bruto, necessitando de uma limpeza e pré-processamento dos dados (Kuhn, 2013). O primeiro passo tomado foi o tratamento de *null values (N/A ou missing values)*, dados que se encontram em branco, gerados pelo não preenchimento pelos usuários. Existem diversas formas de tratar os *N/A*, e a forma que foi utilizada neste projeto leva em conta o tamanho do banco de dados em questão, que possui observações e variáveis em grande quantidade. É importante salientar que, qualquer forma de tratamento dos *missing values* vai alterar o banco de dados e de certa forma, criara um viés. Existem muitas possíveis causas de um valor em branco, como já dito, eles, em grande parte, são dados pelo não preenchimento por conta dos clientes, mas isso pode se dar por vários motivos. O individuo pode não querer preencher, pois se trata de algo que ele pode considerar prejudicial ao seu *score*, e a retirada desse valor implica que não estamos

levando em conta esse fato. Porem, pode ser apenas um erro computacional, ou, uma variável que apresenta muitos *null values,* pode não ter sido formulada de forma adequada ao publico alvo (Sicsu, 2012).

Não foi possível encontrar em nenhum livro relacionado ao tema a proporção que seria aceitável de *null values* em uma variável. Isso iria variar em relação ao quão importante qualitativamente ela é para o modelo. Porem, como dito anteriormente, não foi fornecida uma legenda para classificar as variáveis. Em um fórum de discussão dentro de um site que trabalha com o tema de ciência de dados (Analytics Vidhya), contribuintes deram a sugestão de adotar um corte entre 30 e 50%, ou seja, se a variável apresentar uma proporção de *missing values* maior do que o corte, ela deveria ser retirada do modelo. Com isso, foram retiradas mais de 40 variáveis que apresentavam ocorrências excessivas de valores em branco.

Além dos dados internos, coletados pela própria instituição financeira, também foram oferecidos dados de um *bureau* de crédito, uma empresa externa que mantem dados sobre inadimplência de pessoas físicas, ela tem acesso a empréstimos passados, possíveis protestos, entre outras informações relevantes para a modelagem de risco de crédito. Porém, foi disponibilizado se cada individuo já fez algum requerimento de crédito dentro do último ano, trimestre, mês, semana, dia e hora. E, para diminuir o número de variáveis usadas no modelo, optei por manter o maior intervalo de tempo, que engloba todas as outras variáveis. Sendo assim, mantive apenas a variável binaria que indica se o cliente já fez um pedido de crédito dentro do último ano.

Por último, optei de forma arbitraria retirar a variável referente ao sexo do cliente. No Brasil, não existe uma legislação que proíbe o uso dessa variável em modelos de crédito, cabendo ao desenvolvedor tomar a decisão de usar ou não. Porem, em outros países, como nos Estados Unidos, não é permitido a utilização. Sendo assim, a variável foi descartada do banco de dados.

Como já dito anteriormente, além da análise quantitativa das variáveis apresentadas, uma grande parte do tempo do analista de crédito é tomado pela análise qualitativa dos dados apresentados (Sicsu, 2012). No entanto não foi possível realizar estre processo, e decidi manter o restante dos dados, para que não acabe retirando nenhum dado sensível para o modelo em questão.

Após a retirada dos dados considerados descartáveis, foi feito o tratamento das variáveis que iriam se manter no algoritmo final. Dados qualitativos foram transformados em *dummy variables*, uma forma de quantificar dados categóricos. E com os dados quantitativos analisei se eles seguem uma distribuição monotonica, e caso não tenham, seria importante dicretizar a variável, ou seja, pegar

uma variável quantitativa e transformar em categorias (Sicsu, 2012). Importante dizer que para redes neurais não é necessário fazer a discrtização, pois elas lidam melhor com a não linearidade, já se tratando de regressão logística, é necessário realizar esse processo. Sendo assim, separei em dois *datasets* diferentes, um para cada modelo. Após todos os passos, o banco de dados pronto para a aplicação dos modelos consistia em 49 variáveis e 260 mil observações.

4.2 Modelagem e Teste

Com os dados em mãos, prontos para serem usados, utilizei o *package* "h2o" para o desenvolvimento da rede neural. Este pacote possibilita a criação de diversos tipos de estruturas de redes neurais, e como *Credit Scoring* é uma aplicação relativamente pouco demandante de processamento, seria suficiente para sua modelagem. No entando este package poderia ser usado para objetivos mais complexos, já que você consegue ativar quantas camadas e *nodes* você achar necessário, e conta com as funções de ativação mais usadas. Além disso, ele possibilita a utilização de todos os *threads* do processador, que seriam núcleos de processamento virtuais que possibilitam a divisão do trabalho dado ao computador, e consequentemente a obtenção de resultados com mais rapidez, algo que o R por si só não possibilita. No caso deste projeto, só tenho em disposição dois, porem, em computadores mais modernos, o trabalho poderia ser dividido em mais segmentos. A únicas desvantagens desse package é que ele não possibilita a utilização do *GPU* (placa gráfica) como fonte de processamento, o que deixaria o processo de treinamento ainda mais eficiente.

O modelo de regressão logística também foi feito em R, utilizando a função "glm", que já vem pré-instalada com o *package* de estatística básico do software. Porem, diferente da rede neural, após a primeira vez que foi "rodado" o modelo, é possível realizar a seleção de variáveis utilizando o método *stepwise backward*, o que irá descartar mais algumas categorias que o processo não considerar importantes. Este método de seleção começa com todas as variáveis do modelo, e vai retirando as que não influenciam de forma positiva na precisão da regressão.

Por fim, os dois modelos foram comparados usando as mesmas métricas. Primeiro foi testado a capacidade de separação entre bons e maus pagadores, utilizando o teste de AUROC (*Area Under Receiver Operation Curve*). Para isso, também foi utilizado o software R, mais especificamente o *package* "hmeasure". Este teste calcula integral definida da curva que representa a capacidade do modelo em dividir a variável dependente em suas possíveis categorias, utilizando diversos pontos

de corte. E seus resultados se encontram entre 0,5 e 1, sendo 0,5 um modelo que não tem capacidade de discriminação, e 1, um modelo que separa perfeitamente a amostra. Ou seja, ele analisa a eficácia do modelo como todo, não em um ponto de corte específico, o que torna ele mais efetivo do que outros métodos muito usados na área, como o KS (índice de Kolmogorov-Smirnov), que é o mais utilizado no mercado brasileiro (Sicsu, 2012).

A última métrica utilizada foi o teste de Spiegelhalter. Com o objetivo de estimar a calibração do modelo, ou seja, se ele gera as probabilidades de um cliente ser um bom ou mau pagador de forma eficaz. O *package* utilizado dentro do R foi o "rms". E o teste consiste em verificar se o valor do Escore de Brier (BS) é suficientemente grande para aceitar a Hipótese Nula, e consequentemente, o modelo estima bem as probabilidades.

$$BS = \frac{1}{N}\sum_{1}^{N}(di - PD_i)^2$$

di – 1 = se i for mau pagador; 0 = se i for bom pagador

PD_i – Probabilidade estimada de ser mau pagador

N – Tamanho da amostra teste

BS = 0, modelo estima perfeitamente

BS = 1, o modelo não tem capacidade preditiva

5 RESULTADOS

5.1 Rede Neural

O modelo de rede neural utilizado foi um relativamente simplista, já que, como falado anteriormente, o problema abordado não requer uma grande capacidade computacional, a qual poderia ser entregue utilizando arquiteturas mais complexas (SICSU, 2018). Então, foi utilizada uma *Shallow Neural Network*, na qual é utilizada apenas uma camada de processamento. Porem, não existe uma regra específica de quantos *nodes* deveriam ser colocados dentro da camada de processamento. Existem autores que indicam utilizar "p" *nodes* sendo p = (número de inputs), o seja o número de variáveis sendo utilizadas, e ir diminuindo o número ate se chegar a um resultado

ótimo (Shimueli, 2010). Porem, também existem quem fale que a logica deve ser o contrário (Sicsu, 2019). O que podemos extrair é que se trata de tentativa e erro. No caso deste projeto o número utilizado foi 23 *nodes* na única camada de processamento.

Outro parâmetro necessário para o funcionamento da rede neural é a quantidade de *epochs*, isto indica o número de vezes em que os dados são "passados" pelo modelo (Aggarwal, 2018). O valor que será passado para o algoritmo não pode ser nem tão grande, pois no limite, o modelo pode não convergir e tender ao infinito. E nem tão pequeno, pois não teria o treinamento necessário. Sendo assim, mantive 10 como o número usado, que seria o padrão do package. Outro ponto, já tratado anteriormente, foi a função de ativação utilizada, neste caso, a Tahn, que para problemas de classificação binária, usando probabilidades, tende a gerar bons resultados (Aggarwal, 2018).

Sendo assim, após o treino do modelo, realizei as previsões na amostra teste. E medi a capacidade de divisão de amostras do modelo, usando o teste AUROC, chegando em um valor de 0,6623, lembrando que o teste varia de 0,5 sendo um modelo incapaz, e 1, sendo um modelo perfeito. Com esse valor, podemos concluir que a rede neural teve uma capacidade discriminatória aceitável. Podemos também inferir que, esse valor não foi mais alto por conta da proporção muito baixa de um tipo de pagador na amostra (92% bons pagadores e 8% maus aproximadamente), o que interfere na habilidade de qualquer modelo. Para representar o valor encontrado no teste, usa-se uma curva, onde sua área varia de 0,5 e 1. E é uma função da sensitividade e de (1 – especificidade), que são respectivamente medidas que indicam a quantidade de maus clientes classificados corretamente, e bons clientes classificados da forma errada, também chamados de falso alarme (Sicsu, 2012).

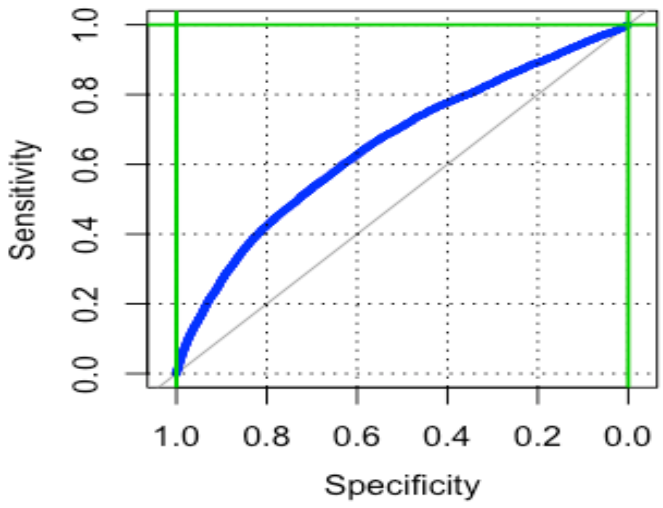

Figura 4 - Produzida pelo autor usando R

O outro teste pelo qual o modelo deve ser submetido é para analisar seu poder de calcular as probabilidades, usando o teste de Spiegelhalter. O método usado no teste seria para desenvolver um teste de hipótese que utiliza o Escore de Brier e ver se rejeitaríamos a hipótese nula, de que o modelo está bem calibrado.

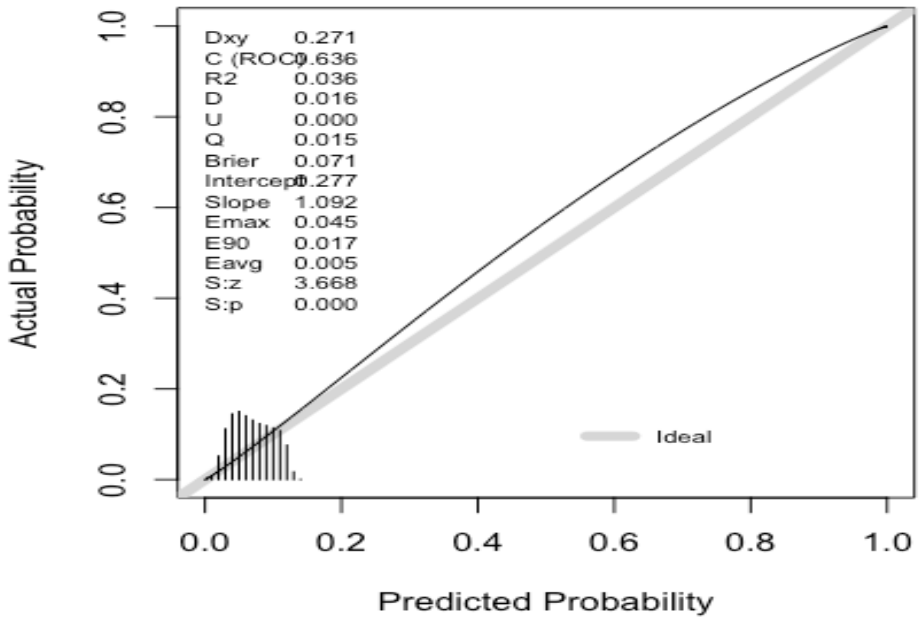

Figura 5 - Criado pelo autor usando R

Com o resultado obtido, podemos concluir que a calibração não foi suficiente, algo que pode ser relacionado com a amostra ser viesada para apenas uma classe, prevendo bem probabilidades baixas, mas mal valores altos. O que pode ser observado no histograma gerado no canto inferior esquerdo da figura 5.

5.2 Regressão Logística

Um individuo, “I”, pode pertencer a dois grupos em um modelo de *credit scoring.* Ou seja, a variável dependente, ou apenas Y, pode tomar dois valores, Y = 1 ou Y = 0. Caso o individuo

pertença ao grupo que Y = 1, ele está no chamado grupo resposta (Sicsu, 2012). E com a regressão logística, podemos estimar a probabilidade de um individuo estar no grupo resposta, P(Y = 1 | I), sendo P(Y = 0 | I) = (1 – (P(Y = 1 | I)).

Agora, a razão entre os quocientes de probabilidade, ou em inglês, *odds*. Pode ser denotada pela seguinte formula:

$$odds = \frac{P(Y = 1)}{1 - (P(Y = 1))}$$

As observações de um banco de dados se relacionam com variáveis, e a regressão logística parte do pressuposto de: (Hosmer, 2000)

$$\ln\left(\frac{P(Y = 1)}{1 - \big(P(Y = 1)\big)}\right) = ß0 + ß1 + ß2 + \cdots + ßn$$

Portanto, o algoritmo treinado em R, irá estimar os coeficientes ($ß0 + ß1 + ß2 + \cdots + ßn$) referentes a cada variável. Sendo que, Z = ß0 + ß1X1 + ß2X2 + ... + ßnXn. Assim, podemos deduzir que a relação entre Z(I) e P(Y = 1) pode ser denotada por:

$$P(Y = 1) = \frac{e^Z}{1 + e^Z}$$

Finalmente, para estimar os coeficientes, foi utilizada a função “glm” do pacote básico do próprio R para rodar o modelo de regressão logística. O valor obtido no teste AUROC foi de 0,7047. O que demonstra que o modelo tem uma capacidade aceitável da divisão entre categorias.

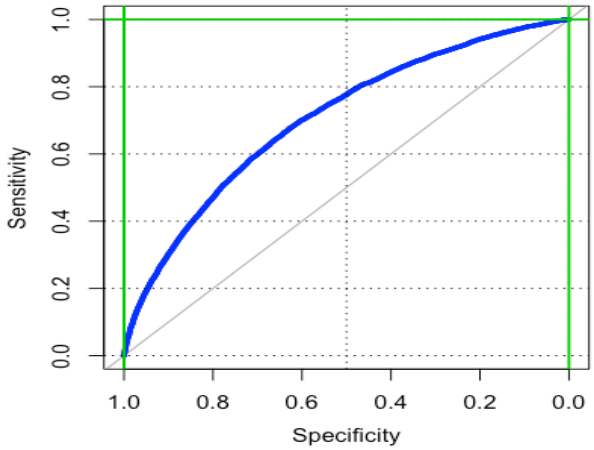

Figura 6 - Criado pelo autor usando R

E o teste de Spiegelhalter, usando um α de 5%, conclui que o modelo não tem uma calibração aceitável. Ou seja, o modelo não estima a probabilidade de *default* de forma confiável.

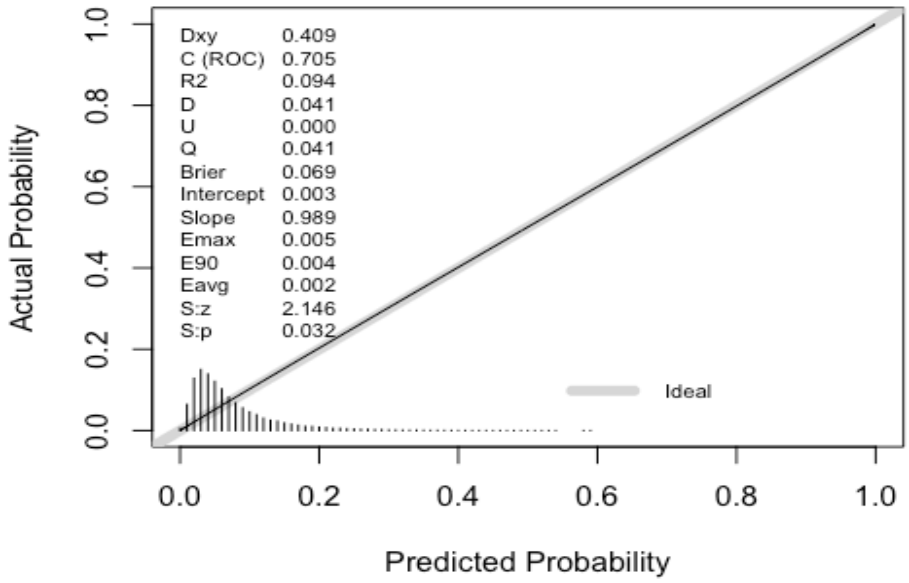

Figura 7 - Criado pelo autor usando R

5.3 Comparação

Como já foi mencionado anteriormente, foi usando o mesmo banco de dados, com a mesma divisão entre treinamento e teste para realizar uma comparação direta. As mesmas variáveis foram usadas, as que foram obtidas após a fase de pré-processamento. Na regressão logística, podemos fazer a seleção de variáveis que são estatisticamente significantes, ou seja, que contribuem de forma positiva no modelo como um todo, e que não estão apenas fazendo volume do banco de dados. Isso se da pela seleção *stepwise*, que pode ser feito de diversas formas. Porem, para atestar qual algoritmo era melhor para essa tarefa específica, não foi utilizado esse artificio, já que não existe algo do gênero para redes neurais.

Sendo assim, primeiro podemos observar o output do teste AUROC. Na rede neural, obtivemos um valor de 0,67, aproximadamente. Já na regressão logística, o valor encontrado foi aproximadamente 0,70. Com isso, concluímos que, evidentemente, o segundo modelo se deu melhor em separar a amostra teste nas suas duas categorias, bom e mal pagador. Porem, a diferença foi relativamente pequena, e pode ter se dado pela capacidade da rede neural em capturar as relações da amostra de treinamento de forma exagerada. Esse algoritmo consegue, muitas vezes "decorar" o banco de dados, assim, quando é aplicado a uma outra amostra ele não consegue encontrar exatamente as mesmas relações. Por esse fato, é necessário um banco de dados suficientemente grande para o treinamento do modelo, além de se ponderar o tempo de treinamento disponibilizado (Aggarwal, 2018). Além disso, os dados usados para esse modelo de *credit scoring*, eram, como já falado, tendenciosos para bons pagadores, uma vez que 92% das observações encontradas na variável dependente são iguais a 0, o que representa os bons pagadores.

Agora, ao analisarmos os resultados do teste de Spiegelhalter para os dois modelos, mais uma vez podemos concluir que a regressão logística se saiu melhor para a previsão da probabilidade de inadimplência dos indivíduos apresentados. Com ela, foi possível chegar mais próximo de aceitar a hipótese nula. Porem, ao se tratar da rede neural, utilizando a mesma confiança, o valor p encontrado no teste não é suficientemente grande para aceitar a hipótese nula. Esse fato, pode ser explicado de novo pela capacidade da rede neural em abstrair os padrões da amostra de treinamento. É possível observar no histograma da figura 5, que as probabilidades estão concentradas em valores baixos. Já, quando analisamos o histograma da figura 7, pela própria natureza da regressão logística, ela tende a não segmentar 100% dos valores dentro de um espectro tão baixo, gerando uma distribuição positivamente inclinada.

6 CONCLUSÃO

Dada todas as possíveis aplicações das redes neurais, e tudo que este modelo de algoritmo vem proporcionando de inovação tecnológica, é inquestionável a capacidade e o potencial que ele guarda. Porém, ao analisarmos tarefas menos complexas, como esta, talvez, toda sua capacidade não acabe contribuindo de forma positiva em todos os casos.

O grande fator, que estabelecera o possível ganho ao utilizar *deep learning* é a quantidade e a qualidade dos dados disponíveis. Neste caso, o banco de dados, apesar de ser suficientemente grande, contando com mais de 300 mil observações e mais de 120 variáveis. A coluna de bons e maus pagadores tinha uma distribuição que faz com que todos os modelos de *machine learning* ou estatísticos sofram, dificultando uma segmentação e previsão precisa. Já que, o *dataset* continha uma proporção de bons pagadores muito maior do que de maus pagadores, o algoritmo acaba ficando tendencioso para apenas um lado.

O que poderia ser feito para minimizar o potencial destrutivo dessa situação, seria utilizar de técnicas de *oversampling* (Kuhn, 2016). Esses métodos tentam analisar como os indivíduos estão distribuídos em relação a suas variáveis. Por exemplo, digamos que uma base de dados apresente muitos adultos com uma renda liquida maior do que de indivíduos mais jovens, o modelo iria criar novas observações não apenas relacionando idade e renda liquida, mas iria tentar capturar todas as relações existentes, e gerar novos clientes com base nisso. E, dessa forma, ao rodarmos um modelo de rede neural, ou qualquer outra forma de previsão, o viés gerado será menor, fornecendo uma capacidade discriminatória e uma calibração mais efetiva. Essa forma de tratamento dos dados só não foi realizada nesse projeto pois requere uma certa capacidade computacional. O que não foi possível alcançar utilizando meu computador.

Métodos estatísticos tradicionais vem sendo usados por muitas décadas, e não por qualquer motivo. Sua capacidade preditiva, como mostrado neste estudo, supera muitos modelos mais sofisticados de inteligência artificial. Com isso, não são necessariamente em todas as tarefas que se compensará o aumento do gasto da demanda computacional apenas para a utilização de *deep learning*. Sempre, deve ser testada ambas as formas, caso o aumento do poder de previsão seja suficientemente maior, aí sim deve-se investir na aplicação, cabendo ao analista realizar o julgamento.

Em todos os campos da ciência é possível reconhecer padrões. Jim Simons, matemático americano e fundador de um dos maiores e melhores fundos de investimentos do mundo, disse, “Nos

procuramos, por meio de dados históricos, padrões anormais que não seriam esperados de ocorrer aleatoriamente". Ele, em seu fundo, Renaissance Technologies, utiliza inferências matemáticas e *machine learning* para gerir investimentos de uma forma nunca antes vista. Esse é um exemplo de aplicação, outras são em modelos de previsão de demanda, modelagem de *churn*, criação de portfolios de investimento, análise de mercado, são todas possíveis aplicações de modelagem preditiva e consequentemente de redes neurais, porem, sempre se deve ficar atendo a disponibilidade dos dados, assim com sua quantidade e qualidade para diferentes algoritmos de modelagem preditiva.

Já é possível observar a utilização com maestria de algoritmos para ajudar o dia a dia de diversos profissionais. Recentemente, só foi possível observar uma foto de um buraco negro com a ajuda de *deep learning* para realizar a decodificação dos dados para formação de uma imagem (Shah, 2019). A Tesla, empresa americana de carros elétricos, comprou em 2019 a DeepScale, companhia com foco em algoritmos de inteligência artificial, baseados em redes neurais, com o intuito de aperfeiçoar e deixar mais compactos seus programas que permitem que os carros vendidos pela empresa consigam dirigir sem o *input* humano (Reisinger, 2019). Portanto, como a tecnologia vem cada vez mais se desenvolvendo e a literatura e a comunidade em geral relacionada ao tema também cresce a cada dia, cabe ao profissional atual pensar nas possibilidades de sua aplicação, facilitando, para quem goste do tema e esteja disposto a estudar, a otimização do seu trabalho.

7 REFERÊNCIAS

Kuhn, M. *Applied Predictive Modeling*. Estados Unidos: Springer, 2016.

Fakkor, R. *Using Deep Learning to Enhance Cancer Diagnosis and Classification*. Estados Unidos: ICML, 2013.

Aggarwal, C. *Neural Networks and Deep Learning*, Estados Unidos: Springer, 2018.

Laredo, A. *Credit Scoring.* São Paulo: Blucher, 2010.

Strang, G. *Introduction to Linear Algebra*, Estados Unidos: Wellesley-Cambridge, 2016.

Géron, A. *Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow,* Estados Unidos: O'Reilly Media, 2019.

Foote, *A Brief History of Deep Learning*, 2017. Disponível em: <https://www.dataversity.net/brief-history-deep-learning/#>. Acesso em: 23/12/2019.

Markoff, *How Many Computers to Identify a Cat? 16,000*, 2012. Disponível em: <https://www.nytimes.com/2012/06/26/technology/in-a-big-network-of-computers-evidence-of-machine-learning.html>. Acesso em: 23/12/2019.

Cambridge Dictionary, *Big Data*. Disponível em: <https://dictionary.cambridge.org/us/dictionary/english/big-data>. Acesso em: 23/12/2019.

Edwards, *Chess grandmaster Garry Kasparov on what happens when machines 'reach the level that is impossible for humans to compete'*, 2017. Disponível em: <https://www.businessinsider.com/garry-kasparov-talks-about-artificial-intelligence-2017-12>. Acesso em: 23/12/2019.

Chandra, *McCulloch-Pitts Neuron – Mankind's First Mathematical Model of A Biological Neuron*, 2018. Disponível em: <https://towardsdatascience.com/mcculloch-pitts-model-5fdf65ac5dd1>. Acesso em: 23/12/2019.

Arunava, *The Perceptron,* 2018. Disponível em: <https://towardsdatascience.com/the-perceptron-3af34c84838c>. Acesso em: 25/05/2019.

McDonald, *Machine Learning fundamentals (II): Neural Networks,* 2017. Disponível em: <https://towardsdatascience.com/machine-learning-fundamentals-ii-neural-networks-f1e7b2cb3eef>. Acesso em: 25/05/2019.

Shuvayan, *What should be the allowed percentage of Missing Values?,* 2015. Disponível em: <https://discuss.analyticsvidhya.com/t/what-should-be-the-allowed-percentage-of-missing-values/2456>. Acesso em: 25/05/2019.

Shah, *When Neural Networks Saw the First Image of a Black Hole*, 2019. Disponível em: <https://medium.com/analytics-vidhya/when-neural-networks-saw-the-first-image-of-black-hole-3205e28b6578>. Acesso em: 31/05/2019.

Reisinger, *Tesla Autopilot AI DeepScale*, 2019. Disponível em: <https://fortune.com/2019/10/02/tesla-autopilot-ai-deepscale/>. Acesso em 31/05/2019.