GUSTAVO HWU LEE

APRENDIZADO DE MÁQUINA PROFUNDO NA ANÁLISE DE SEGMENTAÇÃO DE CLIENTES

Trabalho de Formatura apresentado à Escola Politécnica da Universidade de São Paulo para a obtenção do Diploma de Engenheiro de Produção

São Paulo

2018

GUSTAVO HWU LEE

APRENDIZADO DE MÁQUINA PROFUNDO NA ANÁLISE DE SEGMENTAÇÃO DE CLIENTES

Trabalho de Formatura apresentado à Escola Politécnica da Universidade de São Paulo para a obtenção do Diploma de Engenheiro de Produção

Orientadora:

Prof.ª. Dra. Celma de Oliveira Ribeiro

São Paulo

2018

FICHA CATALOGRÁFICA

AGRADECIMENTOS

Primeiramente, agradeço aos meus pais e irmãos, por todo o suporte, dedicação e valores recebidos durante todos os momentos.

Aos meus colegas da faculdade, com quem compartilhei grande parte dos momentos no dia a dia, e por todo o companheirismo e ajuda durante a graduação.

À professora Celma de Oliveira Ribeiro, por toda ajuda no desenvolvimento das ideias e pela dedicação no desenvolvimento desse trabalho de formatura. Os conhecimentos desenvolvidos durante esse trabalho foram importantes para a minha formação profissional e acadêmica.

Agradeço também os professores da Engenharia de Produção da Poli, que me ajudaram a construir o que sei hoje e a preparar para os desafios do mercado de trabalho.

RESUMO

O propósito desse trabalho é identificar clientes através de modelos de aprendizado de máquina, visando definir estratégias para aumentar o número de clientes e operações. O trabalho será desenvolvido em uma empresa prestadora de serviços de financeiros de transferências internacionais de moeda e câmbio. Em função das peculiaridades dos produtos oferecidos, como as diferentes intenções de realizar uma transferência internacional, seja para o pagamento de um curso ou um tratamento de câncer, o perfil de clientes é muito diversificado.

Para identificar os padrões e classificar os clientes com base em categorias de valor médio das operações, redes neurais com aprendizado profundo (em tradução livre do inglês *deep learning*) foram treinadas com dados históricos e a precisão testada em dados de validação. Diversas tipologias de redes neurais foram comparadas para avaliar a melhor precisão e selecionar o melhor modelo. Diferentes informações foram utilizadas para treinar o modelo, como endereço de residência, idade, gênero, valor financeiro da primeira operação, entre outros.

Foram testados três modelos diferentes, dois deles com os clientes de ambas plataformas isolados e no outro com os clientes combinados. Para cada um dos três modelos, foram testadas diversas arquiteturas de redes neurais, com diferentes quantidades de camadas neurais e neurônios ocultos por camada. O modelo final treinado e calibrado tem como objetivo alimentar as ferramentas de disparos de e-mail e de vendas, nas quais é comum ser necessário segmentar os clientes pelo valor financeiro médio para diferentes estratégias de comunicação e funis de vendas. Os resultados de validação dos modelos identificaram uma boa capacidade de classificação dos clientes corretamente pelo modelo. Por fim, selecionou-se o modelo com melhor precisão e arquitetura para ser implementado na base de dados das ferramentas.

Palavras chave: Segmentação de Clientes. Redes Neurais Profundas. Aprendizagem de Máquina. Avaliação de Precisão de Redes Neurais.

ABSTRACT

Every business must know its customers. The purpose of this study is to predict the average ticket value of each customer based on signup information, website access patterns and sales in an international transfer platform and a money exchange platform. The company has two websites, one for each type of service. The customers of each platforms are very diversified. In the money exchange platform, the main intent is to exchange money for traveling. The customers in the international transfer platform varies a lot, ranging from a father sending money to his son studying abroad to a person seeking for medical services for cancer treatment.

Deep neural networks were trained with historical data to identify patterns and classify customers based on categories of average ticket value. The precisão of the model was tested in a validation set. Many networks architectures were testes and compared against each other to identify the best model. Many types of data were used to train the model such as geolocation information, age, genre, first operation value, and many others.

Three models were tested. Two of them had customers of each platform separate from each other. The third model had customers of both platforms combined into a single model. In each of the three models, many architectures were tested with different amounts of hidden layers and hidden units in each layer. The final model was trained and fine-tuned with the goal to feed e-mail marketing tools and sales tools which is common the need to segment customers by the average ticket values of different communication strategies. The results of the validation of the models identified good precisão on classifying customers correctly. The best model architecture was selected, and the software was integrated with the marketing and sales tools database.

Keywords: Customer Segmentation. Deep Neural Networks. Machine Learning. Neural Network Precisão Evaluation.

LISTA DE FIGURAS

Figura 1 - Topologia de uma rede neural com múltiplas camadas 16
Figura 2 – Separação dos Conjuntos de Dados para o Modelo 17
Figura 3 - Neurônio Artificial em uma rede neural com múltiplas camadas 18
Figura 4 – Função de Ativação Linear 19
Figura 5 – Função de Ativação Sigmoide 20
Figura 6 – Função de Ativação Tangente Hiperbólica 20
Figura 7 - Função de Ativação ReLu 21
Figura 8 – Tipos de Aprendizagem de Máquina Supervisionado e Não Supervisionado 24
Figura 9 – Exemplo dos Algoritmos de Otimização do Método do Gradiente 28
Figura 10 – Efeito da Taxa de Aprendizado no Treinamento 28
Figura 11 – Exemplo de Dropout e Drop Connect 31
Figura 12 - Método de validação por *Holdout* 33
Figura 13 - Método do K-Fold *Cross Validation* 34
Figura 14 – Precisão dos Modelos com Algoritmo de Otimização RMSProp 49
Figura 15 – Precisão dos Modelos com Algoritmo de Otimização Adam 50
Figura 16 – Precisão do Modelo Combinado com *Cross Validation* e Algoritmo de Otimização Adam 55
Figura 17 - Precisão do Modelo Combinado com *Cross Validation* e Algoritmo de Otimização RMSProp 55
Figura 18 – Rede Neural Escolhida 57
Figura 19 – Arquitetura da Integração do Modelo com o *Data Warehouse* 58
Figura 20 – Funcionamento do Atualizador dos Dados pelo Modelo 59

LISTA DE TABELAS

Tabela 1 – Campos da Tabela de Clientes 40
Tabela 2 – Campos da Tabela de Operações 41
Tabela 3 – Campos da Tabela de Visitas 42
Tabela 4 - Configurações Utilizadas no Modelo 43
Tabela 5 - Composição dos Dados de Entradas dos Modelos de Redes Neurais 43
Tabela 6 - Descrição dos Campos de Entrada da Rede Neural 44
Tabela 7 – Tratamento dos Dados em Cada Campo 46
Tabela 8 - Categorias do Modelo da Rede Neural 47
Tabela 9 – Desvio Padrão da Precisão das Arquiteturas do Modelo Câmbio com Otimização Adam 51
Tabela 10 – Desvio Padrão da Precisão das Arquiteturas do Modelo Câmbio com Otimização RMSProp 52
Tabela 11 – Comparativo dos Modelos e Algoritmos de Otimização 52
Tabela 12 – Quantidade de Arquiteturas por Resultado do Teste de Hipóteses 54
Tabela 13 – Dez melhores Precisões das Arquiteturas do Modelo Combinado 54
Tabela 14 – Precisão Final do Modelo Escolhido 56

SUMÁRIO

1. INTRODUÇÃO 11

1.1. Descrição da empresa e contexto do estágio 11

1.2. O problema 13

1.3. Objetivo 13

1.4. Estrutura do trabalho 14

2. REVISÃO BIBLIOGRÁFICA EM REDES NEURAIS 15

2.1. Introdução 15

2.2. Conjuntos de Dados de Validação e Treinamento 16

2.3. O Neurônio 17

2.3.1. Função de Ativação Linear 19

2.3.2. Função de Ativação Sigmoide 19

2.3.3. Função de Ativação Tangente Hiperbólica 20

2.3.4. Função de Ativação ReLu 20

2.3.5. Função de Ativação Softmax 21

2.4. Funções de Perda 22

2.5. Aprendizagem Supervisionada em Redes Neurais Artificiais 24

2.5.1. Método de Aprendizagem 24

2.5.2. Otimização do Método do Gradiente 26

2.5.3. Taxa de Aprendizagem 28

2.5.4. Otimização RMSProp 29

2.5.5. Otimização Adam 29

2.5.6. Momentum 30

2.5.7. Regularização 31

2.6. Avaliação de Precisão 32

2.6.1. Holdout 32

2.6.2. Cross-validation 33

2.7. Pré-processamento dos Dados 35
2.7.1. Dados Numéricos 35
2.7.2. Dados Categóricos 35
2.7.3. Escala dos Dados 36
2.8. Outros Tipos de Redes Neurais 38
3. MODELO EMPÍRICO 39
3.1. Bases de Dados 40
3.1.1. Tabela de Clientes 40
3.1.2. Tabela de Operações 41
3.1.3. Tabela de Visitas 41
3.2. Tratamento dos Dados 43
3.3. Ferramentas Utilizadas 47
3.3.1. Tensorflow 47
3.3.2. Keras 47
3.4. Arquitetura da Rede 47
3.5. Resultado do Treinamento 48
3.6. Integração do Modelo com o Data Warehouse 57
4. CONCLUSÃO 60
APÊNDICE A – RESULTADO DO TESTE DE HIPÓTESES COMPLETO 63
APÊNDICE C – CONSULTAS SQL PARA EXTRAÇÃO DOS DADOS 72
APÊNDICE D – RESULTADOS COMPLETOS DA PRECISÃO DOS MODELOS 76

1. INTRODUÇÃO

O presente trabalho foi desenvolvido em uma empresa de serviços financeiros de câmbio e remessas internacionais, a Beetech. O autor faz parte da área de *Business Intelligence* da empresa, na qual são desenvolvidos diversos relatórios que buscam avaliar o desempenho das áreas por indicadores e identificar oportunidades para a empresa. A área apresenta uma demanda intensa por dados, sendo necessário o constante desenvolvimento do *data warehouse* (banco de dados com informações tratadas e consolidadas) para suprir as demandas por informações dos clientes internos e alimentar as ferramentas das áreas de marketing e comercial.

A área utiliza diferentes modelos para a avaliação de precisão e já implementou modelos de aprendizagem de máquina para a área de marketing para melhorar a precisão dos canais de conversão de clientes. A área conta com um engenheiro de dados para tratar os dados provenientes dos ambientes transacionais, que podem não estar na forma mais adequada para as análises.

O desenvolvimento de indicadores e relatórios exigem um rigor na validação das informações antes de ser apresentado para determinada área, uma vez que impacta diretamente na credibilidade da área e pode resultar em decisões tomadas de maneira errada. A proposta do trabalho é a de identificar os clientes com um maior potencial de gerar receita para a empresa através de modelos matemáticos de aprendizado de máquina, mais especificamente redes neurais de aprendizado profundo.

1.1. Descrição da empresa e contexto do estágio

A Beetech é uma fintech brasileira criada em 2015 que atua na área de câmbio. A empresa começou com a Beecâmbio, uma plataforma online para venda de moedas em espécie e cartões pré-pagos para viagem. O cliente pode optar por retirar o produto em uma das lojas parceiras ou receber em casa através de um portador com seguro no produto. Com uma experiência maior na área, o grupo começou a atuar na área de transferências internacionais com a plataforma Remessa Online, que serve tanto para envio quanto recebimento de moeda estrangeira. A partir desses dois produtos, a empresa expandiu o foco de pessoas físicas para incluir pessoas jurídicas e programa de parcerias. No modelo de parcerias, uma empresa parceira pode revender os produtos da Beetech tanto de câmbio quanto de remessa e receber um comissionamento com

base na quantidade vendida. A empresa também conta com uma área comercial que atende os públicos de pessoa jurídica, parceiros e clientes *private*.

Os clientes utilizam a plataforma da Beecâmbio, sobretudo, quando vão viajar para o exterior e necessitam comprar moeda estrangeira. Os clientes da Remessa Online, em contrapartida, são mais diversificados. As remessas internacionais apresentam um maior espectro de finalidades. A Circular 3.690 de 16/12/2013 do Banco Central dispõe as diferentes naturezas de operações de câmbio, com mais de 100 categorias distintas. Por exemplo, há a categoria de compra ou venda de imóveis no exterior, disponibilidade no exterior, empréstimo, aluguel de imóveis, importação ou exportação de mercadorias, serviço, entre outras.

As principais fontes de receita da empresa são:

- Envios internacionais (Remessa Online): o serviço atende clientes institucionais e pessoas físicas. É destinado para pessoas que desejam enviar dinheiro para uma conta no exterior em moeda estrangeira. Os clientes de pessoa jurídica e pessoa física que opera altos valores são atendidos pela mesa comercial, enquanto os demais clientes realizam o fluxo pela própria plataforma.
- Recebimentos do exterior (Remessa Online): o serviço é utilizado por clientes que querem receber dinheiro do exterior. Os clientes de pessoa física realizam o fluxo pela plataforma, enquanto os clientes institucionais realizam operações pela mesa comercial.
- Serviços de Câmbio (BeeCâmbio): os serviços de câmbio são destinados principalmente para pessoas físicas que necessitam de moedas estrangeiras em espécie. O público predominante são pessoas que vão viajar para algum país. O cliente fecha a cotação do câmbio na plataforma e pode optar por retirar o dinheiro em uma das lojas parceiras ou recebem em casa por um portador.

A estrutura da empresa está organizada em tecnologia, operações, comercial, marketing, financeiro, conteúdo, design e *business intelligence*. Esta última é a área na qual foi realizado o estágio supervisionado pelo autor.

1.2. O problema

A falta de uma forma de segmentação que identifique rapidamente os clientes com maior potencial de gerar receita desafiou o autor a buscar uma forma de quantificar o potencial de gerar valor financeiro com base em modelos de aprendizado de máquina. Essas segmentações são fundamentais para o desenvolvimento das estratégias de comunicação e atendimento comercial dependendo do tipo do cliente.

É fundamental nesse contexto utilizar de listas dinâmicas que filtram os clientes com base em determinadas características. Essas segmentações geradas podem ser programadas para disparos de e-mail marketing ou funis de atendimento comercial em plataformas de CRM (*customer relationship management*). Como a quantidade de pessoas em cada área é limitada, tornando inviável o atendimento por mesa comercial de todos os clientes da plataforma, é fundamental identificar os clientes mais atraentes para a estratégia adotada.

As plataformas acumularam grandes quantidades de dados históricos dos clientes, que podem ser utilizados em algoritmos de aprendizado de máquina para identificar padrões nos dados e identificar os clientes mais atrativos dependendo da estratégia.

Haja vista a importância de segmentar os clientes para elaborar estratégias direcionadas para determinados públicos, a grande quantidade de dados disponíveis e a grande relação com a geração de receita da empresa, o autor decidiu aprofundar seus conhecimentos em aprendizado de máquina, mais especificamente, redes neurais de aprendizado profundo para identificar os clientes com maior potencial de gerar receitas.

1.3. Objetivo

O objetivo desse trabalho é identificar os clientes com maior potencial financeiro visando a definição de estratégias de atendimento comercial e de campanhas de marketing para aumentar a receita da empresa. Os clientes preenchem uma série de informações para permitir o cadastro na plataforma, de modo que há muitas informações que podem auxiliar na determinação do potencial financeiro do cliente.

O modelo proposto para determinar o potencial financeiro do cliente busca prever a categoria de receita média por operação a partir de dados que já podem ser obtidos a partir da primeira operação na plataforma, permitindo uma identificação rápida do cliente sem que seja necessário esperar um período coletando dados para determinar o potencial financeiro do cliente.

Serão utilizados dados como localização residencial, limite de operação aprovado, valor da primeira operação, padrões de navegação e outros como entrada para o modelo determinar a classificação do cliente.

As redes neurais de aprendizado profundo são aproximadores universais de funções e requerem um conjunto de dados para treinar o modelo. Pretende-se elaborar diferentes modelos e arquiteturas de redes neurais testando diferentes algoritmos de otimização de treinamento e, para cada uma delas, avaliar a precisão para escolher o melhor modelo para ser implementado. Por fim, o a arquitetura final da rede escolhida será implementada e integrada às principais ferramentas de disparo de e-mail e CRM (*customer relationship management*), na qual estão os funis de atendimento comercial.

1.4. Estrutura do trabalho

No Capítulo 2 é apresentado o referencial teórico sobre rede neurais com as metodologias utilizadas para avaliar precisão e seleção de modelos

No Capítulo 3, os conceitos apresentados no Capítulo 2 são aplicados para desenvolver um modelo de segmentação de clientes. Para tanto, são apresentados as bases de dados e os resultados dos modelos testados. Também é discutido brevemente como o modelo será implementado na infraestrutura de banco de dados da empresa.

Por fim, no Capítulo 4, são propostas algumas melhorias para continuidade do presente trabalho e os impactos na empresa.

2. REVISÃO BIBLIOGRÁFICA EM REDES NEURAIS

2.1. Introdução

As redes neurais artificiais são a base da aprendizagem profunda, elas são um modelo matemático de aprendizagem máquina. Segundo Géron (2017), as redes neurais são versáteis, poderosas e escaláveis, tornando-as ideais para tratar de problemas complexos, como classificar bilhões de imagens como no Google Imagens, reconhecer comandos de voz como a Siri da Apple ou recomendar o próximo vídeo como é o caso do Youtube e outras plataformas de mídia.

Ainda de acordo com o autor, as redes neurais foram introduzidas em 1943 pelo neurofisiologista Warren McCulloch e pelo matemático Walter Pitts em "*A logical calculus of the ideas immanent in nervous activity*". Os autores apresentaram um modelo computacional simplificado de como os neurônios biológicos poderiam funcionar utilizando lógica proposicional. Esse modelo foi o pioneiro, desde então foram inventadas diversas novas arquiteturas.

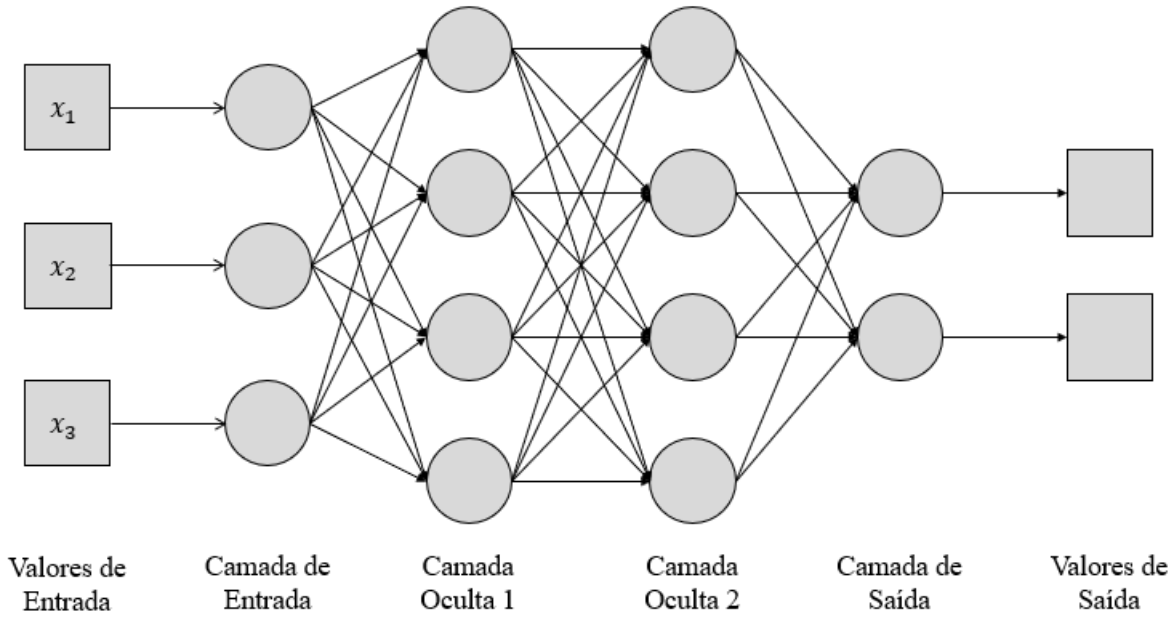
Patterson e Gibson (2017) definem redes neurais como modelos computacionais que compartilham algumas semelhanças com um cérebro animal, no qual pequenas unidades trabalham em paralelo sem um controle centralizado. As redes neurais artificiais são compostas de unidades, também conhecidas como neurônios, que, em conjunto, constituem camadas da rede. Cada neurônio de uma camada intermediária ou oculta está ligado a outros neurônios das camadas anteriores e posteriores, como é possível observar graficamente na Figura 1, que representa a topologia de uma rede neural artificial *feed forward*. Cada círculo é uma unidade da rede, que recebe valores de entrada e devolve valores em sua saída.

Conforme Buduma (2015), essas redes são capazes de representar qualquer função dada uma quantidade suficiente de neurônios. Problemas que envolvem apenas funções lineares podem ser expressos em uma rede neural sem camadas ocultas, apenas com a camada de entrada e a de saída.

Segundo Géron (2017), os modelos de aprendizado de máquina podem ser classificados em quatro categorias principais de acordo com a quantidade e tipo de supervisão recebida: os modelos supervisionados, não supervisionados, semisupervisionados e de aprendizagem por reforço (em tradução livre do inglês *Reinforcement Learning*). As redes neurais são um caso de modelo supervisionado, no qual é necessário fornecer dados com as soluções desejadas para o modelo. Ao contrário do modelo supervisionado, o modelo não supervisionado não recebe os

resultados desejados a partir dos dados de entrada, de modo que o modelo tenta aprender sem que valores de resposta sejam apresentados para o modelo.

Figura 1 - Topologia de uma rede neural com múltiplas camadas



Fonte: Adaptado de Patterson e Gibson (2017)

Já os algoritmos semisupervisionados, segundo Géron (2017), lidam com dados parcialmente classificados com respostas, sendo muito utilizado em algoritmos de imagem. A aprendizagem por reforço, em contrapartida, é baseada em um "agente" que interage com um ambiente, podendo observar o contexto e realizar ações, recebendo recompensas ou penalidades dependendo das ações escolhidas. A partir dessas ações, o agente dentro do ambiente irá determinar a melhor estratégia para determinado objetivo a partir de diversas simulações no ambiente. O modelo foi utilizado para vencer o campeão mundial Lee Sedol no jogo Go, foram utilizadas diversas partidas passadas para treinar o modelo para definir a melhor política (em tradução livre do inglês *policy*) dentro do ambiente do jogo para vencer.

2.2. Conjuntos de Dados de Validação e Treinamento

As redes neurais precisam ser treinadas para ajustar o modelo aos dados, no entanto, esses modelos têm a tendência de sobre ajustar aos dados de treinamento, o que reduz sua precisão quando testada contra dados mais gerais que não estavam no conjunto de treinamento do modelo. Na Seção 2.6 serão detalhadas as técnicas utilizadas para a avaliação da precisão de uma rede e evitar o sobreajuste aos dados do conjunto de treinamento.

Um dos desafios no processo de treinamento do algoritmo é saber quando se deve parar o treinamento para que o modelo não fique muito específico para os dados de treinamento e não generalize bem para um conjunto de dados que não foi visto antes. Para identificar o sobreajuste, é comum separar um conjunto de dados que não é utilizado no treinamento para avaliar a precisão do algoritmo nesse conjunto de dados.

Segundo Buduma (2015), havia um consenso de se distribuir os dados de treinamento e teste na proporção de 80% e 20%, respectivamente. No entanto, com o aumento drástico da quantidade de dados disponíveis e aumento da capacidade de processamento, é possível ter divisões diferentes que não prejudiquem o conjunto de teste, desde que a quantidade de amostras seja suficiente.

A Figura 2 demonstra como os dados devem ser separados. O conjunto de dados original é separado em um de treinamento e um de teste. O conjunto de teste não é utilizado no treinamento do modelo e serve de avaliação final para determinar se houve ou não sobreajuste. O conjunto de treinamento, em contrapartida, é utilizado no processo de aprendizagem do algoritmo, sendo subdividido novamente para avaliar a precisão durante o treinamento.

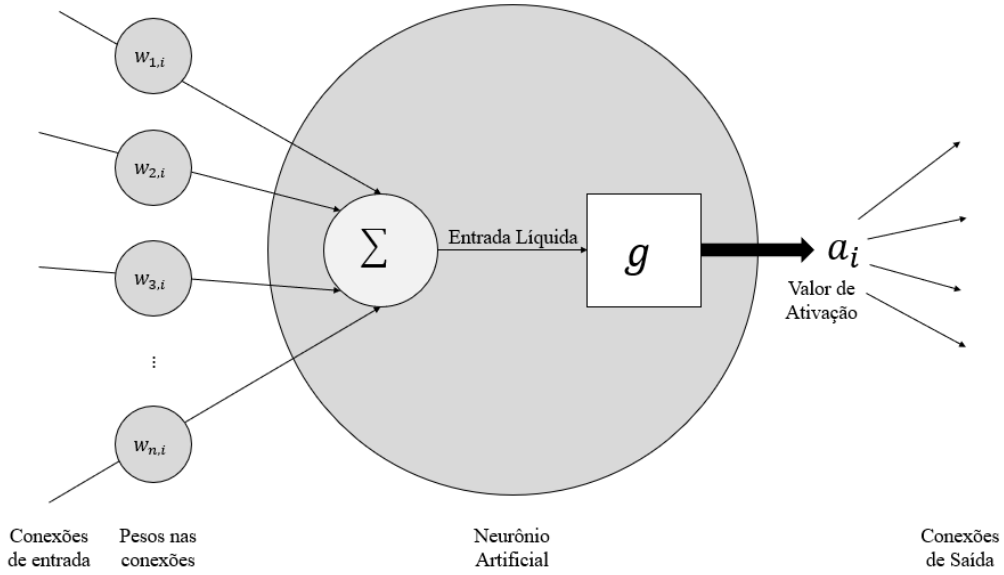
Figura 2 – Separação dos Conjuntos de Dados para o Modelo

Fonte: Adaptado de Raschka e Mirjalili

2.3. O Neurônio

Cada um dos neurônios de uma rede recebe diversas conexões dos outros neurônios da camada anterior que podem ser ignoradas ou não dependendo do peso atribuído à conexão. A função de ativação g na Figura 3 permite obter diferentes resultados da saída do neurônio dependendo da intenção da rede. A função de ativação g irá receber o valor obtido pela multiplicação dos pesos com os valores de entrada da rede e devolverá um valor que depende da intenção do neurônio, de modo que há diferentes tipos de funções de ativação para diferentes redes neurais.

Figura 3 - Neurônio Artificial em uma rede neural com múltiplas camadas



Fonte: Adaptado de Patterson e Gibson (2017)

Conforme Patterson e Gibson (2017), a entrada líquida de um neurônio e os valores de ativação da saída de uma rede são definidos pelas seguinte equações:

$$z_i = entrada\_líquida_i = W_i . A_i$$

$$a_i = g(entrada\_líquida_i)$$

W_i é um vetor de pesos do neurônio i

A_i é um vetor de valores de ativação para o neurônio i

$g(x)$ é uma função de ativação

Para cada camada da rede neural, é necessário acrescentar um termo de *bias* denotado por b. Esse termo é um valor escalar responsável por garantir que, pelo menos, alguns nós de uma rede sejam ativados independentemente do nível do sinal de ativação. Esse termo também é atualizado constantemente no aprendizado da rede neural e permite que a rede tente novas interpretações e comportamentos no caso de um baixo sinal de ativação. Desse modo, considerando o termo de *bias*, as equações reescritas são:

$$z_i = entrada\_líquida_i = W_i . A_i + b$$

$$a_i = g(W_i . A_i + b)$$

a_i é um valor escalar da saída de apenas um neurônio

W_i é um vetor de pesos do neurônio i

A_i é um vetor de valores de ativação para o neurônio i

g é uma função de ativação

b é o termo de bias e é um valor escalar

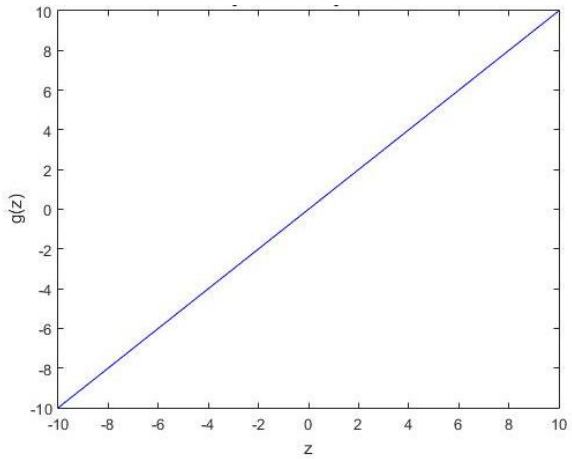
As saídas dos neurônios estão associadas a funções de ativação, que são utilizadas para propagar essa saída de uma camada na rede neural para a próxima camada. A saída do neurônio é a ativação do neurônio. Em uma rede neural com camadas ocultas, as funções de ativação são utilizadas para capacitar a rede neural a representar funções não lineares. A entrada e a saída da função são valores escalares.

2.3.1. Função de Ativação Linear

Na função de ativação linear, a variável dependente é diretamente proporcional à variável independente, de modo que a função passa o sinal sem ser modificado.

$$g(z) = Wz$$

Figura 4 – Função de Ativação Linear



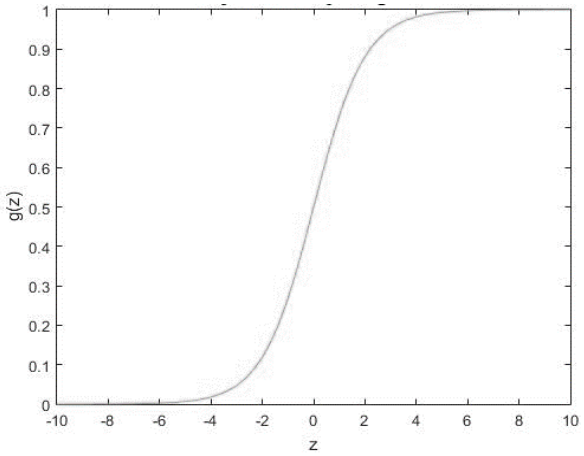
Fonte: Elaborado pelo autor

2.3.2. Função de Ativação Sigmoide

A função de ativação sigmoide transforma variáveis reais (z ϵ $\mathbb{R}$) em probabilidades entre 0 e 1.

$$g(z) = \frac{1}{1 + e^{-z}}$$

Figura 5 – Função de Ativação Sigmoide



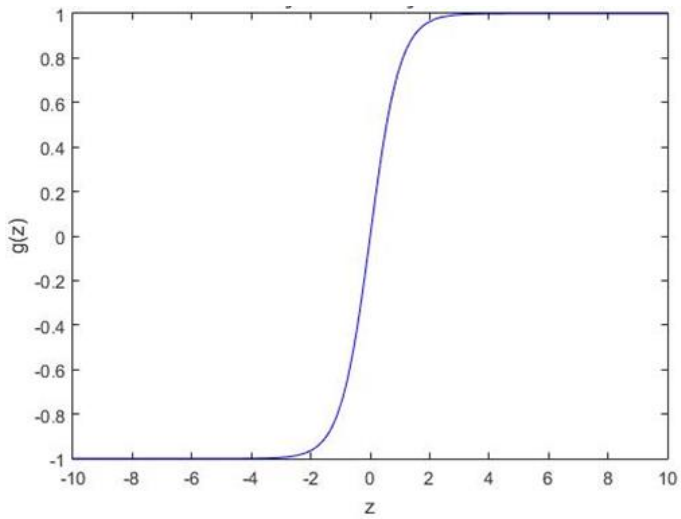
Fonte: Elaborado pelo Autor

2.3.3. Função de Ativação Tangente Hiperbólica

A função de ativação tangente hiperbólica transforma variáveis reais ($z \in \mathbb{R}$) em probabilidades entre -1 e 1. A função tangente hiperbólica é preferida em relação à função de ativação sigmoide, pois está centrada em 0.

$$g(z) = \tanh(z)$$

Figura 6 – Função de Ativação Tangente Hiperbólica



Fonte: Elaborado pelo Autor

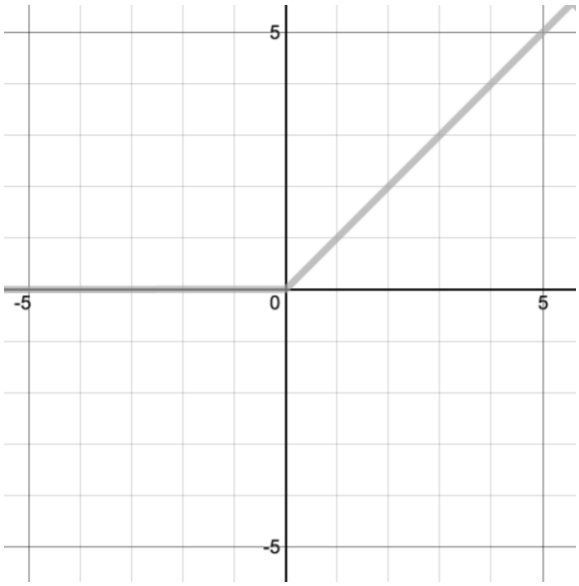
2.3.4. Função de Ativação ReLu

Segundo Patterson e Gibson (2017), a função de ativação ReLu só é ativada quando a entrada da função é maior que um determinado nível. Por ser computacionalmente fácil de

resolver, a função de ativação reduz o tempo de treinamento e converge mais rápido. Para valores negativos, a função se anula e, caso contrário, estabelece uma relação linear com a variável dependente, conforme indica a Figura 7. A função é expressa como:

$$f(x) = \max(0, x)$$

Figura 7 - Função de Ativação ReLu



Fonte: Adaptado de Patterson e Gibson (2017)

2.3.5. Função de Ativação Softmax

O mesmo autor também apresenta a função de ativação Softmax para casos em que se deseja obter um vetor com a distribuição de probabilidade para um conjunto de categorias mutuamente exclusivas. Nesse caso, o vetor resultante deve ser tal que:

$$\sum_{i}^{n} p_i = 1$$

$$P = [p_0\ p_1\ p_2\ \dots p_{n-1}]$$

n é o número de categorias

A normalização das variáveis independentes pode ser obtida através da seguinte função:

$$p_i = g_i(z_i) = \frac{e^{z_i}}{\sum_j^n e^{z_j}}$$

2.4. Funções de Perda

As funções de perda são responsáveis por quantificar o desempenho de uma rede neural em relação à aprendizagem com os dados de treinamento. O erro é calculado com base nas previsões da rede. Os erros de cada previsão são agregados e ponderados para obter a nota final da rede, que é utilizada para avaliar seu desempenho. As funções de perda podem ser classificadas em diferentes finalidades, como regressão e classificação.

Na regressão, a função perda definida pelo erro quadrático médio (MSE) é frequentemente utilizada para obter valores reais como resultado da rede neural. Ao considerar-se um processo de treinamento com M exemplos cada um com P características de entrada e N características de saída na rede, a função perda pelo erro quadrático é dada por:

$$\boldsymbol{L(W,b) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{M}\sum_{j=1}^{M}\left(\hat{y}_{ij} - y_{ij}\right)^2}$$

N é a quantidade de exemplos de entradas na rede

M é a quantidade de características observadas na saída da rede

$\hat{y}_{ij}$ é o valor previsto pela rede

y_{ij} é o valor observado da função

Apesar da função perda pelo erro quadrático médio ser utilizada nos casos mais gerais, é muito suscetível a outliers, de modo que, dependendo da situação a ser analisada, pode ser mais vantajoso utilizar funções que reduzam os efeitos dos outliers. Algumas funções de perda alternativas que podem ser utilizadas para regressão são:

Função perda pelo erro absoluto médio

$$L(W,b) = \frac{1}{2N}\sum_{i=1}^{N}\sum_{j=1}^{M}\left|\hat{y}_{ij} - y_{ij}\right|$$

Função perda pelo erro logarítmico quadrático médio

$$L(W,b) = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M}\left(log\left(\hat{y}_{ij} - log\left(y_{ij}\right)\right)\right)^2$$

Função perda pelo erro absoluto médio percentual

$$L(W,b) = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M}\frac{\left(\mathbf{100} \times |\widehat{\boldsymbol{y}}_{\boldsymbol{ij}} - \boldsymbol{y}_{\boldsymbol{ij}}|\right)}{\boldsymbol{y}_{\boldsymbol{ij}}}$$

Na classificação, por sua vez, segundo Géron (2017), são utilizadas com maior frequência as funções de classificação de *Hinge Loss*, *Logistic Loss* e *Negative log likelihood*. A Hinge Loss é muito utilizada em classificadores fixos, como um classificador binário com resultado fixo de 0 ou 1. A Logistic Loss, por sua vez, é mais utilizada quando as probabilidades são de maior interesse que um resultado fixo, como a probabilidade de pertencer ou não a um grupo, por exemplo.

No caso da classificação binária, é possível utilizar um neurônio com função de ativação sigmoidal. Para a previsão de probabilidades, como é o caso da Logistic Loss, é necessário garantir que a soma das probabilidades para cada resultado seja igual a um, sendo necessário um neurônio com a função de ativação *softmax* na última camada. Conforme Patterson e Gibson (2017), quando trata-se de multiplicações de probabilidades, é comum converter em somas do log das probabilidades, de modo a obter a função perda do *Negative log likelihood*.

Hinge Loss

$$\boldsymbol{L(W,b)} = \frac{\mathbf{1}}{\boldsymbol{N}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} \boldsymbol{max}(\mathbf{0}, \mathbf{1} - \boldsymbol{y}_{\boldsymbol{ij}} \times \widehat{\boldsymbol{y}}_{\boldsymbol{ij}})$$

Logistic Loss:

$$\boldsymbol{L(W,b)} = \prod_{\boldsymbol{i=1}}^{\boldsymbol{N}} \widehat{\boldsymbol{y}}_{\boldsymbol{i}}^{\boldsymbol{y_i}} \times (\mathbf{1} - \widehat{\boldsymbol{y}}_{\boldsymbol{i}})^{\mathbf{1}-\boldsymbol{y_i}}$$

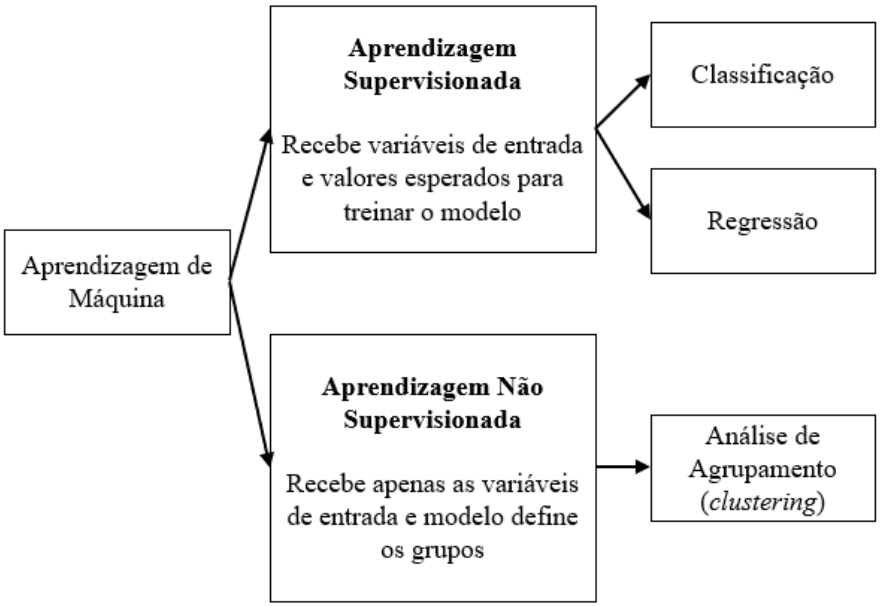
Negative log likelihood

$$\boldsymbol{L(W,b)} = -\sum_{\boldsymbol{i=1}}^{\boldsymbol{N}}\sum_{\boldsymbol{j=1}}^{\boldsymbol{M}} \boldsymbol{y}_{\boldsymbol{i,j}} \times \boldsymbol{log}(\widehat{\boldsymbol{y}}_{\boldsymbol{ij}})$$

2.5.Aprendizagem Supervisionada em Redes Neurais Artificiais

Segundo Géron (2017), as redes neurais são um caso de aprendizagem supervisionado, de modo que são apresentados nos dados de treinamento as variáveis de entrada e a saída esperada do modelo. Desse modo, as principais tarefas realizadas por uma rede neural do tipo *feed forward* são regressão e classificação. A Figura 8 apresenta algumas das tarefas realizadas pelos modelos supervisionados e não supervisionados.

Figura 8 – Tipos de Aprendizagem de Máquina Supervisionado e Não Supervisionado



Fonte: Adaptado de Géron (2017)

2.5.1. Método de Aprendizagem

De acordo com Géron (2017), o aprendizado em uma rede neural ocorre pelo mecanismo de retro propagação (em tradução livre do inglês backpropagation) que visa minimizar o erro de uma função de perda, que pode ser de regressão ou classificação. O objetivo desse processo é determinar os parâmetros w que minimizam esse erro. O algoritmo consiste em uma etapa de passo para frente (*forward pass*) e outra de passo para trás (*backward pass*). Na primeira etapa o algoritmo calcula o resultado a partir das entradas de treinamento na rede. Caso o resultado esteja correto, nada ocorre. Caso contrário, na etapa de passo para trás, os pesos dos neurônios são atualizados a partir do gradiente da função de perda que está sendo minimizada.

Para exemplificar o algoritmo da retropropagação responsável por determinar os parâmetros w, serão utilizados neurônios com funções de ativação sigmoidal e uma função de perda de erro quadrático. O valor resultante y de uma rede com neurônio de saída com função de ativação sigmoidal é dado por:

$$z = \sum_k w_k x_k$$

$$y = g(z) = \frac{1}{1 + e^{-z}}$$

A partir do resultado, é necessário calcular o gradiente da função erro em relação aos pesos w.

Primeiro, calcula-se as derivadas parciais em relação aos pesos:

$$\frac{\partial z}{\partial w_k} = x_k$$

$$\frac{\partial z}{\partial x_k} = w_k$$

$$\frac{dy}{dz} = \frac{e^{-z}}{(1 + e^{-z})^2} = y(1 - y)$$

Em seguida, aplica-se a regra da cadeira para obter a derivada parcial do resultado y em relação aos pesos:

$$\frac{\partial y}{\partial w_k} = \frac{dy}{dz}\frac{\partial z}{\partial w_k} = y(1 - y)x_k$$

A partir das derivadas parciais, é possível calcular a derivada parcial da função erro em relação a cada peso. Como exemplo, será utilizado a função do erro quadrático:

$$E = \frac{1}{2}\sum_i \left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

y é o valor verdadeiro

$\hat{y}$ é o valor obtido pela rede neural

Desse modo, é possível calcular a derivada parcial do erro em relação a cada peso:

$$\frac{\partial E}{\partial w_k} = \sum_i \frac{\partial E}{\partial y^{(i)}}\frac{\partial y^{(i)}}{\partial w_k} = -\sum_i x_k^{(i)}\hat{y}^{(i)}(1 - \hat{y}^{(i)})(y^{(i)} - \hat{y}^{(i)})$$

Concluindo, a regra para a atualização dos pesos pelo método do gradiente é dada por:

$$\Delta w_k = \sum_i \alpha x_k^{(i)} \hat{y}^{(i)}(1 - \hat{y}^{(i)})(y^{(i)} - \hat{y}^{(i)})$$

α é o coeficiente de aprendizado

O coeficiente de aprendizado controla o tamanho de cada passo de iteração, de modo que deve ser ajustado para convergir com uma velocidade controlada para o ponto de mínimo da função quadrática neste exemplo. Algumas das dificuldades no processo de convergência e técnicas para otimizar essa taxa de aprendizado serão discutidos na Seção 2.5.3, Seção 2.5.4 e Seção 2.5.5.

Para aplicar o algoritmo de retro propagação, é necessário treinar não só um neurônio, como no exemplo, mas múltiplas camadas de neurônios. O método primeiramente descrito por Rumelhart, Hinton e Williams (1986) permite analisar a velocidade da variação do erro com base nas alterações dos pesos de uma conexão individual, de modo que é possível achar o caminho de maior inclinação para o ponto ótimo. Por se tratar de um espaço de multidimensional, cada unidade pode afetar muitas unidades de saída, de forma que as derivadas parciais de uma camada oculta são utilizadas para calcular as derivadas parciais da camada anterior. A fórmula de atualização dos pesos para uma rede com múltiplas camadas é dada por:

$$\Delta w_{ij} = \sum_k \hat{y}_i^{(k)} \hat{y}_j^{(k)} \left(1 - \hat{y}_j^{(k)}\right) \frac{\partial E^{(k)}}{\partial y_j^{(k)}}$$

k pertence ao conjunto de dados de treinamento

i refere-se à camada da rede neural

j refere-se à camada anterior à camada i

2.5.2. Otimização do Método do Gradiente

O método do gradiente é o mais utilizado para a otimização de redes neurais. Conforme Ruder (2016), há três principais variantes do método o *batch gradient descent, stochastic gradient descent e o mini-batch gradient descent.* A principalo diferença entre eles é a quantidade de dados utilizados para computar o gradiente da função objetivo. Dependendo da quantidade de dados, há um *trade off* entre precisão e tempo de treinamento.

No *batch gradient descent*, o gradiente da função de perda é calculado pelo conjunto de dados inteiro, de modo que é necessário computar todo o conjunto de dados para realizar apenas uma atualização dos pesos conforme a equação:

$$W = W_{iteração\ anterior} - \alpha\ \nabla_{\mathrm{w}} L(W, b, X, Y)$$

X refere-se aos valores de entrada do conjunto de dados de treino completo
Y refere-se ao valor resultante do conjunto de dados de treino completo

Segundo o autor, uma das desvantagens dessa técnica de otimização é que, para conjuntos de dados grandes, são realizados muitos cálculos redundantes, pois utiliza-se o mesmo dado visto antes para cada atualização do gradiente.

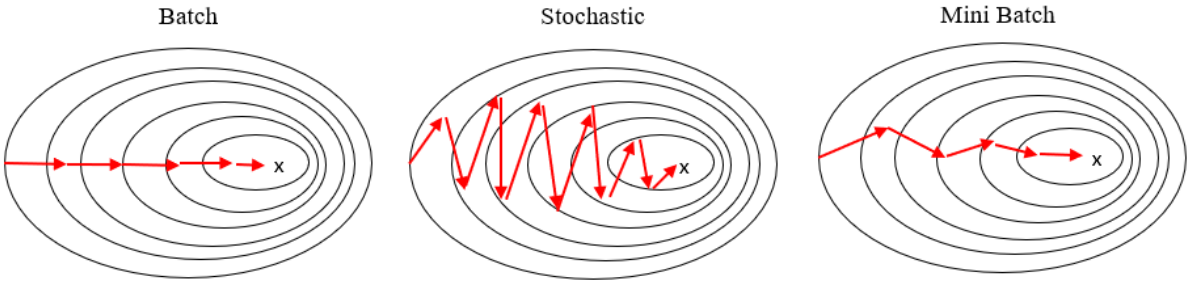
No *stochastic gradient descent*, em contrapartida, para cada um dos dados presentes no conjunto de treinamento, o gradiente é atualizado uma vez, o que torna o treinamento muito mais rápido, mas aumenta a flutuação dos valores da função objetivo. Uma das vantagens dessa variação é permitir que se encontre mínimos melhores para a função objetivo. A técnica pode ser vista na equação:

$$W = W_{iteração\ anterior} - \alpha\ \nabla_{\mathrm{w}} L\left(W, b, x^{(i)}, y^{(i)}\right)$$

$x^{(i)}$ refere-se ao dado i presente no conjunto de treino
$y^{(i)}$ refere-se ao valor resultante do dado i presente no conjunto de treino

O *mini-batch gradient descent*, por sua vez, é uma combinação das duas técnicas anteriores, de modo que a cada atualização um conjunto de *n* exemplos é utilizado para atualizar os pesos. De acordo com Ruder (2016), as principais vantagens desse método são reduzir a variância da atualização dos parâmetros e utilizar de operações matriciais otimizadas em lote. Essa técnica geralmente é a preferida entre as três. A Figura 9 exemplifica os três algoritmos de otimização.

Figura 9 – Exemplo dos Algoritmos de Otimização do Método do Gradiente

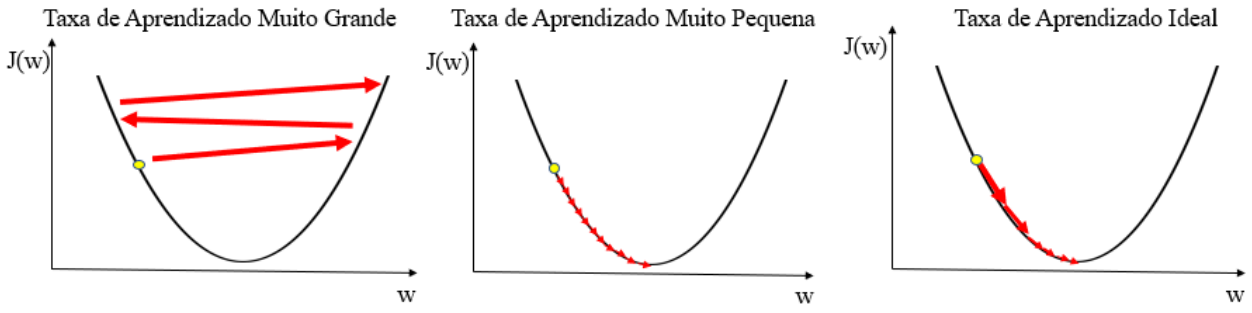


Fonte: Adaptado de Géron (2017)

2.5.3. Taxa de Aprendizagem

Segundo Buduma (2015), a taxa de aprendizado α é a velocidade com que os pesos são atualizados em direção ao ponto ótimo. Há uma dificuldade de se determinar a taxa de aprendizado ideal. Se o valor for muito grande, o treinamento pode não convergir, pois, apesar de no começo ir rapidamente em direção ao ponto ótimo, ao chegar mais próximo, irá pular o ponto ótimo devido ao valor de α. Caso contrário, quando a taxa é muito pequena, o algoritmo irá convergir para o ponto ótimo, mas pode demorar muito tempo. A Figura 10 apresenta esse comportamento da variação da função perda próximo à região ótima em relação a diferentes taxas de aprendizado.

Figura 10 – Efeito da Taxa de Aprendizado no Treinamento



Fonte: Adaptado de Géron (2017)

Como o valor ideal da taxa de aprendizado é específico para cada conjunto dados, não há metodologia trivial para definir os hiperparâmetros ideias, apesar de haver algoritmos de otimização que serão discutidos a seguir.

2.5.4. Otimização RMSProp

Conforme Patterson e Gibson (2017), o algoritmo de otimização RMSProp é um modelo de otimização da taxa de aprendizado muito eficiente, mas não está publicado na literatura, de modo que a maior parte dos que utilizam em artigos acadêmicos citam o slide 29 da aula 6 do curso no Coursera de Geoff Hinton. Em diversos artigos presentes na literatura, como o de Roy (2017), sobre reconhecimento de caracteres por aprendizado profunda e diversos livros publicados como "*Hands-On Machine Learning with Scikit-Learn & TensorFlow*" e "*Python Machine Learning*", observa-se esse tipo de citação, de modo que não há um artigo formal publicado pelo autor do algoritmo. O aprendizado dos pesos da rede neural pode não ser adequado quando a taxa de aprendizado α está muito alta e os pesos oscilam entre o ponto ótimo de convergência demorando muito para alcançá-lo.

Segundo Roy (2017), o método consiste de otimização em manter uma média móvel do quadrado do gradiente para cada peso e dividir pela raiz desse valor. O parâmetro de ajuste da taxa de aprendizado para cada peso é dado por:

$$r_t = (1-\gamma)L'(W_t)^2 + \gamma r_{t-1}$$

L' é a derivada da função perda no momento da iteração t
γ é a taxa de decaimento

Desse modo, cada peso é atualizado por:

$$W_{t+1} = W_t - \frac{\alpha}{\sqrt{r_t}} L'(W_t)$$

α é a taxa de aprendizado
r_t é o parâmetro de ajuste da taxa de aprendizado

2.5.5. Otimização Adam

De acordo com Kingma (2015), Adam é um algoritmo de otimização estocástica baseada no método do gradiente. Adam é considerado computacionalmente eficiente e requer pouca memória, o que reduz o tempo de aprendizado nas atualizações da taxa α de aprendizado. O algoritmo atualiza as médias móveis do gradiente (m_t) e do quadrado do gradiente (v_t),

enquanto os hiper-parâmetros $\beta_1, \beta_2 \epsilon\ [0,1)$ controlam o decaimento exponencial dessas duas médias móveis.

A regra de atualização da taxa de aprendizado do algoritmo consiste em:

$$\Delta_t = \alpha \frac{\widehat{m}_t}{\widehat{v}_t}$$

$\widehat{m}_t$ é a média móvel do gradiente

$\widehat{v}_t$ é o quadrado do gradiente

α é a taxa de aprendizagem do algoritmo

No entanto, há dois limites superiores que devem ser respeitados na atualização:

$$|\Delta_t| \leq \frac{\alpha\ .\ (1-\beta_1)}{\sqrt{1-\beta_2}}, quando\ (1-\beta_1) > \sqrt{1-\beta_2}$$

$$|\Delta_t| \leq \alpha, caso\ contrário$$

2.5.6. Momentum

O uso de momento como forma de otimização foi primeiramente proposto por Polyak (1964) em "*Some methods of speeding up the convergence of iteration methods*". Diferentemente do método do gradiente, que a cada iteração irá atualizar os pesos em pequenos passos regulares, a adição de momento irá acelerar o processo de convergência ao considerar a modificação de uma iteração anterior na interação presente. Para tanto, acrescenta-se um parâmetro adicional na atualização dos pesos. Conforme Badea (2014), o parâmetro acelera o processo de convergência da rede neural em regiões planas ou reduz o efeito de "pulos" em regiões com muitas variações ao adicionar uma fração da modificação anterior na interação presente. Desse modo, a atualização dos pesos de cada neurônio recebe esse parâmetro adicional conforme a seguinte equação:

$$\Delta w(n+1) = -\alpha \frac{\partial E}{\partial w} + m\Delta w(n)$$

m é $a\ taxa\ de\ momento$

$\Delta w(n+1)$ é $a\ mudança\ do\ peso\ na\ iteração\ n+\ 1$

$\Delta w(n)$ é $a\ mudança\ do\ peso\ na\ iteração\ n$

α é $a\ taxa\ de\ aprendizado$

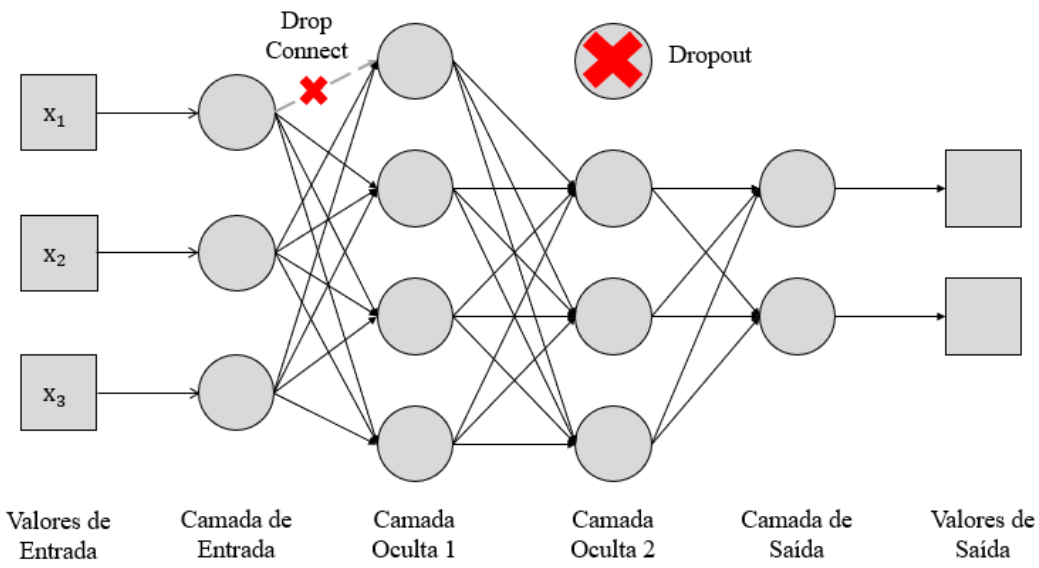
$\frac{\partial E}{\partial w}$ é $a\ deriavada\ parcial\ da\ funçaõ\ erro\ em\ relação\ a\ w$

2.5.7. Regularização

Segundo os autores, a regularização é uma medida para evitar o sobreajuste do modelo aos dados. O sobreajuste ocorre quando o modelo é bom em descrever os dados de treinamento, mas não generaliza bem em dados novos. A regularização dos hiperparâmetros ajuda a modificar o gradiente para que não vá em direções que se sobre ajustam aos dados.

O Dropout e o Drop Connect, por exemplo, são técnicas que consistem em desativar parte da entrada da rede, de modo que a rede neural irá aprender em outros conjuntos de neurônios da rede neural. Isso faz com que a rede neural aprenda representações mais genéricas e não dependa de apenas um determinado valor de entrada. No caso do Dropout, um neurônio da camada oculta é removido de forma aleatória, de modo que não irá contribuir no processo de aprendizado por retro propagação. O Drop Connect, em contrapartida, em vez de desativar por completo o neurônio, desativa-se apenas uma ligação entre dois neurônios. A Figura 11 ilustra como são aplicadas essas técnicas em uma rede neural.

Figura 11 – Exemplo de Dropout e Drop Connect



Fonte: Elaborado pelo Autor

De acordo com Géron (2017), as regularizações L1 e L2 previnem que o espaço de parâmetros torne-se muito grande em apenas uma direção, acrescentando uma penalidade para

pesos muito grandes. Para tanto, são adicionadas penalidades nas funções de perda proporcionais aos pesos, de modo que a função de perda final é igual à função de perda original acrescida de uma penalidade para pesos grandes, conforme as seguintes equações:

$$Regularização\ l_1\ \ E = L(W,b) + \lambda \sum_{j=0}^{M} |W_j|$$

$$Regularização\ l_2\ \ E = L(W,b) + \lambda \sum_{j=0}^{M} W_j^2$$

M é a quantidade de exemplos de treino na rede
$L(W,b)$ é a função perda
λ é o coeficiente de regularização

A regularização L1 multiplica o valor absoluto dos pesos em vez de seus quadrados, o que transforma muitos dos pesos em 0 e permite que apenas alguns aumentem. Isso torna mais fácil a interpretação dos pesos, reduzindo o sobreajuste. A regularização L2, em contrapartida, consiste na adição de um termo na função objetivo da rede que reduz os quadrados dos pesos em vez do valor absoluto. Isso melhora a generalização e suaviza os resultados com mudanças nos valores de entrada da rede. O parâmetro λ de regularização é o responsável pela troca entre encontrar um bom ajuste e manter o peso para certas variáveis da rede

2.6. Avaliação de Precisão

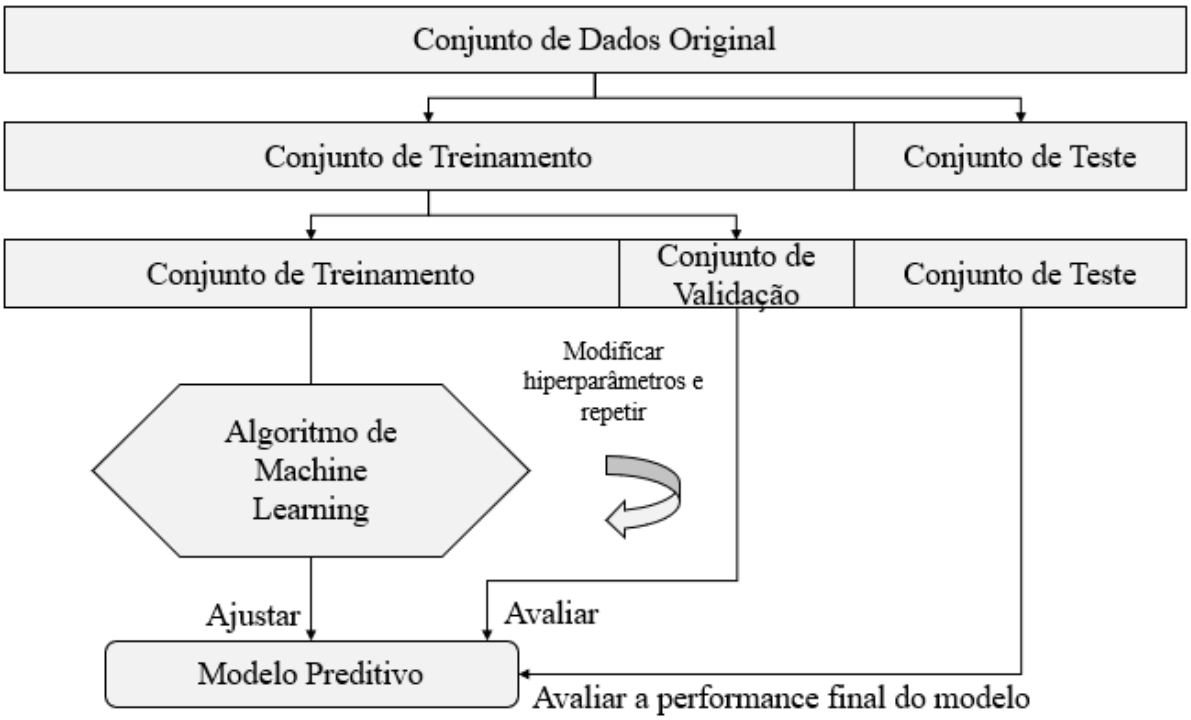
Segundo Raschka e Mirjalili (2017), uma das etapas fundamentais na construção de modelos de aprendizado de máquina é estimar a precisão em um conjunto de dados no qual o modelo ainda não foi treinado. Um modelo pode se ajustar demais aos dados de treinamento se estiver complexo demais ou se ajustar menos que o ideal se for muito simples em relação à complexidade dos dados.

2.6.1. Holdout

Ainda segundo Raschka e Mirjalili (2017), o método por *holdout* de validação do modelo é o mais popular e comum para generalizar a precisão de um modelo de aprendizado de máquina. O método consiste em dividir os dados iniciais em um conjunto de dados para

treinamento e validação. O primeiro conjunto é utilizado para treinar o modelo, enquanto o segundo é utilizado para determinar qual a capacidade de generalização do modelo. É comum no desenvolvimento de modelos de aprendizado de máquina o ajuste de hiperparâmetros para melhorar a capacidade do modelo em dados desconhecidos. Se no processo de seleção do melhor modelo forem utilizados o mesmo conjunto de dados de treinamento, é mais provável que o modelo esteja sobre ajustado para esse conjunto específico de dados. Apesar desse problema, muitos utilizam esse método para a seleção de modelos, o que não é uma boa prática. A Figura 12 ilustra o funcionamento do método.

Figura 12 - Método de validação por *Holdout*



Fonte: Adaptado de Raschka e Mirjalili (2017)

Uma grande desvantagem desse método é que a precisão do algoritmo é muito sensível à forma como os dados foram separados em treinamento e teste, de modo que a precisão irá variar dependendo de como os dados foram particionados.

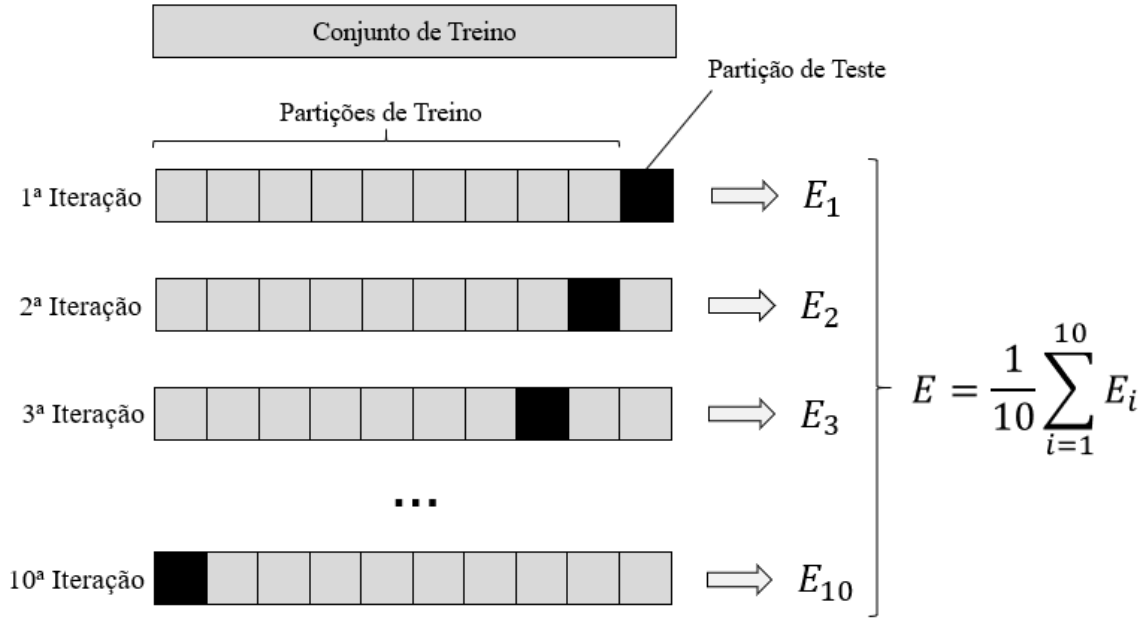
2.6.2. Cross-validation

Segundo os autores, o método de avaliação de precisão por K-fold cross-validation é muito mais robusto quando comparado ao anterior. O método consiste em repetir o método de *holdout* k vezes em k subconjuntos dos dados de treinamento. O conjunto de treinamento é aleatoriamente particionado em k subconjuntos, de modo que k-1 conjuntos são utilizados para

treinamento e o outro para avaliar a precisão do modelo. Esse processo é repetido k vezes, de modo a se obter k modelos distinto e estimativas de precisão.

Por fim, de acordo com Xu (2018), a avaliação final do modelo consiste na precisão média de cada um dos treinamentos em subconjuntos independentes. Isso resulta em uma estimativa muito menos sensível à forma como os dados de treinamento são sub particionados quando comparado ao método por *Holdout*. Como o método consiste em uma sub amostragem sem reposição, cada um dos pontos no conjunto de dados será utilizado para utilizado para treinamento e validação, o que gera em uma menor variância do resultado da avaliação que o modelo anterior. A Figura 13 exemplifica o conceito do método com 10 subconjuntos.

Figura 13 - Método do K-Fold *Cross Validation*



Fonte: Adaptado de Raschka e Mirjalili

No exemplo, o conjunto de dados é particionado em 10 subconjuntos e durante as 10 iterações 9 subconjuntos são utilizados para treinamento e 1 para validação. A estimativa da precisão E é obtida pela média da E_i estimativas de precisão. Um valor padrão para k é 10, com base em resultados empíricos realizados por Kohavi (1995) em diversos conjuntos de dados reais. Para conjuntos de dados pequenos, no entanto, pode ser útil aumentar a quantidade de subconjuntos, de modo que um maior conjunto de dados de treinamento será utilizado a cada treinamento.

2.7.Pré-processamento dos Dados

Segundo Raschka e Mirjalili (2017), nas aplicações no mundo real, é comum faltar dados por várias razões como erros no processo de coleta, alguns dados não serem aplicáveis ou o campo ter sido deixado em branco, como em uma pesquisa, por exemplo. A maior parte das ferramentas de aprendizado de máquina são incapazes de lidar com os dados faltantes ou geram resultados inesperados. Para a modelagem prática com dados reais, há uma série de técnicas para lidar com dados faltantes como remover linhas de dados ou completar com dados de outro conjunto de dados.

Essa etapa de tratamento dos dados é essencial para o bom funcionamento do algoritmo de treinamento, uma vez que os processos anteriores descritos para a regularização não terão efeito adequado caso os dados não estejam normalizados e tratados.

2.7.1. Dados Numéricos

Uma das técnicas apresentadas pelos mesmos autores consiste na completa remoção das entradas com dados faltantes. Essa abordagem, apesar de conveniente, pode acabar removendo muitos dados do conjunto e tornar a análise menos confiável. Desse modo, é proposta uma série de técnicas baseadas em interpolação para o preenchimento de dados faltantes. Para dados numéricos, a substituição pela média dos valores existentes é o mais comum. Alternativas também utilizadas são a substituição dos dados faltantes pelos valores mais frequentes ou pela mediana.

2.7.2. Dados Categóricos

Raschka e Mirjalili (2017) também propõem formas de tratar dados categóricos, que podem ser categorias nominais e ordinais. Nas categorias de nominais não há uma relação de sequência entre as categorias como cores de uma camiseta, enquanto, nas ordinais, há uma relação de sequência como tamanhos de camiseta (L > M > P). No caso dos dados ordinais, é necessário transformar as categorias em números para permitir que o algoritmo seja capaz de identificar a relação de sequência dessa classificação.

Exemplo de transformação de dados ordinais:

$\boldsymbol{Cor}$	$\boldsymbol{Tamanho}$	$\boldsymbol{Preço}$		$\boldsymbol{Cor}$	$\boldsymbol{Tamanho}$	$\boldsymbol{Preço}$
$Verde$	P	38	→	$Verde$	1	38
$Azul$	M	40		$Azul$	2	40
$Vermelho$	G	50		$Vermelho$	3	50

Um erro bastante comum, segundo o autor, é tentar aplicar a mesma técnica para categorias nominais, como nas cores, por exemplo.

Exemplo de transformação de dados nominais **incorreta**:

Cor	***Tamanho***	***Preço***		***Cor***	***Tamanho***	***Preço***
Verde	*P*	38	→	1	1	38
Azul	*M*	40		2	2	40
Vermelho	*G*	50		3	3	50

Nesse caso, ao atribuir valores às cores (verde = 1, azul = 2 e vermelho = 3), o algoritmo entende que há uma relação de que o vermelho é maior que o azul e verde; enquanto o azul é maior que o verde. O algoritmo pode gerar resultados úteis mesmo com essa transformação incorreta, mas não serão ótimos, uma vez que cores de camisetas não tem uma relação de ordem.

Uma alternativa para esse problema, segundo Raschka e Mirjalili (2017), é o método do *One Hot Encoding*, que consiste em criar colunas "dummy" para cada um dos valores únicos da categoria nominal.

Cor	***Tamanho***	***Preço***		***Tamanho***	***Preço***	***Azul***	***Verde***	***Vermelho***
Verde	1	38	→	1	38	0	1	0
Azul	2	40		2	40	1	0	0
Vermelho	3	50		3	50	0	0	1

Segundo Raschka e Mirjalili (2017), é necessário atentar que o *One Hot Encoding* introduz multicolinearidade, que pode gerar problemas em alguns métodos que exigem a inversão de matrizes, pois são matrizes computacionalmente difíceis de inverter, o que pode gerar estimativas instáveis. O autor propõe remover uma das colunas para reduzir a correlação entre variáveis, o que não traria nenhum prejuízo às informações presentes. Se a coluna azul fosse removida, por exemplo, a informação ainda é preservada, pois, se verde = 0 e vermelho = 0, implica que a cor deve ser azul.

2.7.3. Escala dos Dados

O mesmo autor argumenta que a etapa de deixar os dados na mesma escala é fundamental no pré-processamento dos dados e é muitas vezes esquecida. Apenas alguns algoritmos de aprendizagem de máquina são invariantes à escala, como as árvores de decisão e *random forest*.

No entanto, a maior parte dos algoritmos se comportam melhor quando os dados estão na mesma escala. Se uma coluna está em uma escala de 1 a 5 e outra na escala de 1 a 1000000, o algoritmo dará maior importância à segunda coluna, pois está preocupado em reduzir os erros da função perda, que será mais afetada pela segunda coluna.

São propostos dois métodos, pelo autor, para tratar dessas diferenças de escala nos dados: normalização por máximo e mínimo e por desvio padrão. Os dois termos são muitas vezes utilizados com o mesmo sentido em algumas áreas, sendo necessário compreender o sentido pelo contexto. Na maioria dos casos, a normalização de máximo e mínimo refere-se a redimensionar os dados em um intervalo de [0,1], que é um caso de dimensionamento de máximo e mínimo. O valor de cada variável utilizando esse método, pode ser calculado como:

$$x_{norm}^{(i)} = \frac{x^{(i)} - x_{min}}{x_{max} - x_{\min}}$$

x_{min} é *o menor valor de uma determinada variável de entrada*

x_{max} é *o maior valor de uma determinada variável de entrada*

Embora a normalização por máximo e mínimo seja uma técnica bastante utilizada quando os valores precisam estar em um intervalo delimitado, o método de normalização por desvio padrão é melhor para a maior parte dos algoritmos de aprendizagem de máquina, sobretudo, os que otimizam uma função, como os que utilizam o método do gradiente. Muitos algoritmos inicializam os pesos iniciais zerados ou com pequenos valores muito próximos a zero.

Ao utilizar o método da normalização por desvio padrão, centraliza-se os dados de cada característica na média 0 com desvio padrão 1, de modo a representar uma distribuição normal e facilitar o aprendizado dos pesos. O método também preserva informações importantes como *outliers*, que são perdidos pelo método de máximo e mínimo. As variáveis podem ser normalizadas pela seguinte equação:

$$x_{std}^{(i)} = \frac{x^{(i)} - \mu_x}{\sigma_x}$$

μ_x é *a média de uma variável de entrada*

σ_x é *o desvio padrão da variável de entrada*

2.8.Outros Tipos de Redes Neurais

O modelo de rede neural apresentado anteriormente é utilizado para regressões e classificações, trata-se de uma rede neural de aprendizado profundo do tipo *feed forward*. As redes neurais também podem ser utilizadas para tarefas mais complexas como a classificação de imagens e voz, por exemplo, mas que não estão contemplados nas seções anteriores.

As redes neurais convolucionais (em tradução livre do inglês *Convolutional Neural network*) e as redes neurais recorrentes (em tradução livre do inglês *Recurrent Neural Network*) são exemplos dessas redes mais complexas.

Segundo Tsaia, Taoa e Yudia (2017), a rede neural convolucional, tornou-se popular nas tarefas de reconhecimento de voz e imagem, a rede possui camadas que realizam convoluções e servem de filtro para identificar características como contornos, cantos e mudanças de cores em imagens, por exemplo. Essas redes também apresentam camadas de *pooling*, que servem para reduzir a dimensão dos dados, uma vez que, tratando-se de imagens, a quantidade de variáveis é muito grande, pois cada pixel da imagem corresponde a um valor de entrada.

Segundo Routraya e Kanungo (2012), as redes neurais recorrentes são uma classe de redes neurais na qual há conexões cíclicas, na qual o resultado de um neurônio é repassado como dado de entrada do próprio neurônio na próxima iteração. Essa característica cria um estado interno do neurônio que permite criar comportamentos temporais dinâmicos.

3. MODELO EMPÍRICO

O modelo empírico tem por objetivo aplicar o modelo de Redes Neurais de aprendizado profundo do tipo *feed forward* em uma aplicação real na empresa, iniciando-se na coleta de dados até a disponibilização do sistema em um serviço na nuvem com atualização automática para integrar com os sistemas de funis de vendas na plataforma de CRM (*customer relationship management*) e disparos de e-mail.

O primeiro passo para o modelo consiste na coleta dos dados, para tanto, é necessário compreender como os dados estão organizados na empresa e quais as informações disponíveis para a análise. Nesse sentido, as bases de dados serão apresentadas na Seção 3.1. Compreendida a organização das informações, é necessário tratar os dados para adequar aos algoritmos. Essa etapa é fundamental para o bom funcionamento do algoritmo. Esquecer de normalizar algum dado, como o valor financeiro da primeira operação, por exemplo, por gerar grandes distorções nos resultados, uma vez que será atribuído um peso muito maior para essa variável. Antes de iniciar a modelagem dos dados, deve-se separar os dados entre dados de treinamento e teste, de modo que o conjunto de teste só deve ser utilizado como uma validação final para avaliar se não houve sobreajuste do modelo, que só pode utilizar os dados de treinamento na etapa de modelagem.

Em seguida, será necessário definir as ferramentas que serão utilizadas para a prototipagem das redes neurais. No caso, optou-se pelo uso do *python* e uma série de bibliotecas de computação científica. Com as ferramentas necessárias, inicia-se a etapa de modelagem. No caso, serão testados três modelos diferentes, um no qual as duas plataformas (remessa e câmbio) estão na mesma rede e outros dois modelos na qual as plataformas estão separadas. Para cada um desses modelos, é essencial determinar os hiperparâmetros, que são o número de camadas, número de neurônios, algoritmo de otimização da taxa de aprendizado, algoritmo de otimização do método do gradiente, funções de ativação, entre outros. Devido à alta quantidade de dados em cenários reais e o alto processamento exigido para treinar as redes neurais, essas técnicas de otimização apresentadas na Seção 0 são cruciais para reduzir o tempo de treinamento das redes.

Dado o papel das técnicas de otimização na redução do tempo de aprendizado, buscou-se testar dois algoritmos distintos para a mesma finalidade. No caso, realizou-se um teste de hipóteses com arquiteturas de redes neurais iguais treinadas com ambos algoritmos para testar se havia diferenças na média da precisão obtida pelos dois algoritmos.

Com os modelos definidos e treinados, é necessário avaliar qual o melhor modelo de rede neural. Para tanto, utilizou-se a metodologia do *k-fold cross validation* apresentada na Seção 2.6.2. Com modelo escolhido, é necessário avaliar se não houve sobreajuste do modelo quando utilizado para classificar um conjunto de dados nunca observado, para tanto, utiliza-se o conjunto de teste. Por fim, é apresentado como será realizada a integração do modelo com a base de dados existente.

3.1. Bases de Dados

Os dados serão extraídos das bases de dados das plataformas de remessa e de câmbio. As informações estão na forma bruta nas bases de dados transacionais e podem ser acessadas por meio de consultas em linguagem SQL.

Ao todo, o banco de dados possui mais de 100 tabelas, sendo grande parte delas essencial para a operação de serviços, como sistemas de validação, calendário de feriados, precificação, entre outros. Para a presente análise, serão utilizadas três tabelas de dados principais e seus dados mais relevantes. (tabela de clientes, operações e visitas).

3.1.1. Tabela de Clientes

A tabela de clientes é responsável pelos registros de dados cadastrais de cada cliente, como cpf, telefone, endereço, gênero, data do último login, plataforma de signup, entre outros campos. É válido ressaltar que essa tabela é compartilhada pelas duas plataformas, tanto o serviço de remessa quanto o de câmbio.

Tabela 1 – Campos da Tabela de Clientes

Campo	Descrição
Plataforma Inscrito	Qual a plataforma o cliente se inscreveu (Câmbio ou Remessa).
Data de Criação	Data que o cliente se cadastrou
Gênero	Masculino ou feminino
Data de nascimento	Data de nascimento do cliente
Endereço	Rua, número do estabelecimento, cidade, estado, cep, país, longitude e latitude.
Limite de Operação	Limite de operação em reais relacionado à declaração de imposto de renda do cliente
Profissão	Profissão do cliente

Fonte: Elaborado pelo Autor

3.1.2. Tabela de Operações

Na tabela de operações, constam os dados de cada operação dos clientes, como a taxa utilizada para fechar o câmbio, a cotação comercial no momento da compra e a situação atual da operação, que pode ser não estar finalizada. É válido atentar que para a análise só serão utilizadas as operações finalizadas, isto é, que o dinheiro já foi transferido ou retirado.

Tabela 2 – Campos da Tabela de Operações

Campo	Descrição
Plataforma da operação	Determina qual foi a plataforma utilizada para realizar a operação Câmbio ou Remessa.
Valor da operação	Valor total em reais da operação já incluindo tributos e taxas
Moeda	Tipo de moeda da operação (dólar americano, euro, libra esterlina, entre outro)
Data de fechamento	Data em que o cliente confirmou a operação
Tipo de operação	Envio de remessa para o exterior, recebimento de dinheiro do exterior, compra de moeda estrangeira, recarga de cartão de viagem
Status da operação	Operação iniciada, aguardando comprovante, aguardando liquidação, finalizada, cancelada, entre outros. É possível segmentar os status em 4 categorias principais: operações inicadas, operações fechadas, operações liquidadas e operações canceladas.
Natureza da remessa	No caso de operações de remessa, há diversas naturezas de operações, como disponibilidade no exterior, conta de investimento, terceiro com vínculo familiar, pagamento de serviços, entre outras possibilidades.

Fonte: Elaborado pelo Autor

3.1.3. Tabela de Visitas

A tabela de visitas é resultante de um serviço de análise de acompanhamento de visitas. O serviço permite obter estatísticas sobre as visitas dos clientes, como tempo na página, quantidade de páginas visitadas e forma que chegou ao site (campanha de marketing, email, parceiros, entre outros).

Tabela 3 – Campos da Tabela de Visitas

Campo	Descrição
Quantidade de Interações	Quantidade de vezes que o cliente interagiu com a página, como acessar um link ou pesquisar.
Quantidade de Ações	Quantidade de ações que o cliente fez na página, o que inclui ações como download e eventos como cadastro ou realizar uma operação.
Quantidade de Visitas	Quantidade de vezes que um cliente visitou a página.
Sistema Operacional	Sistema operacional que o usuário está utilizando: Windows, Mac, iOS, Android, Linux e outros.
Tempo Total da Visita	Duração total da visita em segundos
Site de Origem	Indica de qual plataforma a visita se originou

Fonte: Elaborado pelo Autor

Com o objetivo de avaliar os clientes das plataformas, foram propostos três modelos de redes neurais, um dos modelos combina os clientes da plataforma de câmbio e com o de remessa e os outros dois modelos consideram os clientes das duas plataformas de forma isolada.

A rede neural tem como objetivo prever uma categoria de valor financeiro do cliente com base em dados de cadastro, operações e padrões de acesso nos sites. Para determinar a precisão do modelo, foram treinadas diferentes arquiteturas de redes neurais, variando a quantidade de camadas e a quantidade de neurônio por camadas.

Com vistas a determinar os parâmetros de cada arquitetura, também foram testados dois algoritmos de otimização da taxa de aprendizado: RMSProp e Adam. Em relação ao método do gradiente para treinamento, foi utilizada a técnica do *mini-batch* com conjuntos de dados de 50 clientes por vez (n = 50), de modo a otimizar o tempo de treinamento e reduzir a variância nos valores dos pesos.

Os conjuntos de dados das plataformas foram separados conforme o procedimento comum na literatura, de se utilizar 80% dos dados para treinamento e 20% para a validação final do modelo. As configurações completas utilizadas no modelo podem ser vistas na Tabela 4.

Tabela 4 - Configurações Utilizadas no Modelo

Categoria	Configuração	Valor
Conjunto de Dados	Tamanho do Conjunto de Treinamento	80% dos dados
	Tamanho do Conjunto de Teste	20% dos dados
Aprendizagem	Algoritmo de Otimização do Método do Gradiente	Mini batch gradient descent (n = 50)
	Algoritmo de Otimização da Taxa de Aprendizado	RMSProp e Adam
Arquitetura da Rede	Quantidade de Camadas Ocultas	1 a 4 camadas
	Quantidade de Neurônios por Camada Oculta	1 a 20 neurônios
	Funções de Ativação do Neurônio de Saída	Softmax
	Funções de Ativação dos Neurônios Intermediários	ReLu
	Funções de Ativação dos Neurônios de Entrada	ReLu
	Função Perda	Negative log likelihood
Avaliação de Precisão	Método de Validação	Cross Validation (k = 10)

Fonte: Elaborado pelo Autor

3.2. Tratamento dos Dados

A primeira etapa consistiu em determinar os dados que seriam utilizados para alimentar os modelos. Como a empresa possui duas plataformas separadas, alguns dos dados disponíveis em uma podem não estar presentes na outra plataforma. A Tabela 5 discrimina quais dados foram utilizados em cada um dos três modelos de redes propostas. A descrição detalhada de cada um dos campos pode ser observada na Tabela 6.

Tabela 5 - Composição dos Dados de Entradas dos Modelos de Redes Neurais

Nome do Dado	Modelos		
	Combinado	Plataforma Remessa	Plataforma Câmbio
Limite Aprovado de Câmbio	X		X
Limite Aprovado de Remessa	X	X	
Valor total da primeira operação de remessa	X	X	
Valor total da primeira operação de câmbio	X		X
Cliente possui parceiro associado	X	X	
Data de Nascimento	X	X	X
Sexo	X	X	X
Latitude do endereço	X	X	X
Longitude do endereço	X	X	X

Profissão	X		X
Quantidade de Visitas na Beecâmbio	X		X
Quantidade de Visitas na Remessa	X	X	
Quantidade de Ações Beecâmbio	X		X
Quantidade de Ações Remessa	X	X	
Quantidade de interações Beecâmbio	X		X
Quantidade de interações Remessa	X	X	
Duração da visita na Beecâmbio	X		X
Duração da visita na Remessa	X	X	
Sistema Operacional	X	X	X

Fonte: Elaborado pelo Autor

Tabela 6 - Descrição dos Campos de Entrada da Rede Neural

Descrição dos Dados	**Descrição**
Limite Aprovado de Câmbio	Limite de operação autorizado que o cliente pode operar com base na declaração de imposto de renda
Limite Aprovado de Remessa	Limite de operação autorizado que o cliente pode operar com base na declaração de imposto de renda
Valor total da primeira operação de remessa	Valor total da primeira operação de remessa em reais
Valor total da primeira operação de câmbio	Valor total da primeira operação de câmbio em reais
Cliente possui parceiro associado	Se o cliente tem um parceiro associado ou não. Para saber se o cliente está associado à um parceiro, basta verificar se utilizou algum cupom de um parceiro.
Data de Nascimento	Data de Nascimento
Sexo	Sexo do cliente
Latitude do endereço	Latitude do endereço de residência
Longitude do endereço	Longitude do endereço de residência
Profissão	Profissão do cliente
Quantidade de Visitas na Beecâmbio	Quantidade de visitas única realizadas pelo cliente no site da Beecâmbio
Quantidade de Visitas na Remessa	Quantidade de visitas única realizadas pelo cliente no site da Remessa
Quantidade de Ações Beecâmbio	Quantidade de ações como download e eventos como signup realizados pelo cliente no site da Beecâmbio
Quantidade de Ações Remessa	Quantidade de ações como download e eventos como signup realizados pelo cliente no site da Remessa
Quantidade de interações Beecâmbio	Quantidade interações que o cliente realizou no site da Beecâmbio como visualizações de página e pesquisas.
Quantidade de interações Remessa	Quantidade interações que o cliente realizou no site da Remessa como visualizações de página e pesquisas.
Duração da visita na Beecâmbio	Tempo total das visitas no site da Beecâmbio
Duração da visita na Remessa	Tempo total das visitas no site da Remessa
Sistema Operacional	Sistema Operacional como Windows, Mac, Android, iOS e Linux.

Fonte: Elaborado pelo Autor

A etapa de tratamento dos dados consistiu em identificar dados que necessitavam de tratamentos como normalização e transformação, como ocorre no caso dos dados categóricos. Essas transformações são fundamentais para o funcionamento dos algoritmos da biblioteca Keras utilizada no modelo, uma vez dados não normalizados podem levar a distorções dos pesos, por exemplo. A transformação dos dados categóricos pelo *one hot encoding* também é fundamental, pois o algoritmo é incapaz de interpretar texto puro sem tratamento, como é o caso da variável de profissão.

Algumas variáveis podem não estar disponíveis em determinada plataforma para todos os clientes. A profissão, por exemplo, só é campo obrigatório no site da plataforma de câmbio, de modo que os clientes que se registraram apenas no site da plataforma de remessa não apresentam esse campo preenchido. Os dados de visita também podem não estar presente em alguns clientes, principalmente, por se tratar de uma mecânica baseada no navegador do usuário, que pode falhar por diversas razões dependendo do programa utilizado para acessar o site. Os dados faltantes foram substituídos pela média da variável de entrada para melhorar a precisão do algoritmo. No caso dos dados categóricos, como a profissão e o sistema operacional, foram criadas colunas do tipo "dummy" conforme o método do *One Hot Encoding.* É válido ressaltar que, no caso dos sistemas operacionais, foram criadas variáveis binárias para os sistemas mais comuns como Windows, Mac, iOS e Android. Os dados numéricos também foram normalizados pelo método do desvio padrão, conforme descrito na Seção 2.6.2.

No caso da longitude e latitude, por representarem dados tridimensionais, é necessário mapear as coordenadas esféricas para as coordenadas cartesianas (x, y e z). No caso das coordenadas esféricas extremas, por exemplo, apesar de serem extremas numericamente, as coordenadas representam localizações próximas em termos físicos. Como a maior parte das coordenadas estão no Brasil, não haveria tanto problema, mas em uma situação em que os dados são globais haveria maior distorção dos dados. Desse modo, foram utilizadas as seguintes equações para converter em coordenadas cartesianas.

$$x = \cos(latitude) * \cos(longitude)$$
$$y = \cos(latitude) * \text{sen}(longitude)$$
$$z = sen(latitude)$$

A Tabela 7 detalha cada uma das transformações realizadas em cada variável de entrada da rede neural.

Tabela 7 – Tratamento dos Dados em Cada Campo

Descrição dos Dados	Tratamento dos dados
Limite Aprovado de Câmbio	Normalização por desvio padrão e preenchimento de campos com dados faltantes com a média
Limite Aprovado de Remessa	Normalização por desvio padrão e preenchimento de campos com dados faltantes com a média
Valor total da primeira operação de remessa	Normalização por desvio padrão e preenchimento de campos com dados faltantes com a média
Valor total da primeira operação de câmbio	Normalização por desvio padrão e preenchimento de campos com dados faltantes com a média
Cliente possui parceiro associado	Nenhum tratamento, possui o valor 1 caso tenha parceiro e 0 caso não tenha
Data de Nascimento	Foi calculada a idade da pessoa e, em seguida, aplicou-se a normalização por desvio padrão
Sexo	Foi aplicado o método do *one hot encoding*
Latitude do endereço	Foram transformadas em coordenadas cartesianas
Longitude do endereço	
Profissão	Foi aplicado o método do *one hot encoding*
Quantidade de Visitas na Beecâmbio	Normalização por desvio padrão e preenchimento de campos com dados faltantes com a média
Quantidade de Visitas na Remessa	Normalização por desvio padrão e preenchimento de campos com dados faltantes com a média
Quantidade de Ações Beecâmbio	Normalização por desvio padrão e preenchimento de campos com dados faltantes com a média
Quantidade de Ações Remessa	Normalização por desvio padrão e preenchimento de campos com dados faltantes com a média
Quantidade de interações Beecâmbio	Normalização por desvio padrão e preenchimento de campos com dados faltantes com a média
Quantidade de interações Remessa	Normalização por desvio padrão e preenchimento de campos com dados faltantes com a média
Duração da visita na Beecâmbio	Normalização por desvio padrão e preenchimento de campos com dados faltantes com a média
Duração da visita na Remessa	Normalização por desvio padrão e preenchimento de campos com dados faltantes com a média
Sistema Operacional	Foi aplicado o método do One hot encoding para os principais sistemas operacionais (Windows, Mac, iOS e Android)

Fonte: Elaborado pelo Autor

Ao todo, a quantidade de clientes utilizados presente no conjunto de dados tratado era de 80 mil clientes, sendo que 20% desses clientes foram separados para serem utilizados como um conjunto de teste. Foram calculadas as categorias de valor financeiro médio por operação correspondentes a cada um dos clientes para serem utilizadas no treinamento da rede conforme a Tabela 8.

Tabela 8 - Categorias do Modelo da Rede Neural

Categoria	Descrição
0	Abaixo de R$4000
1	Entre R$4000 e R$7000
2	Entre R$ 7000 e R$10000
3	Acima de R$ 10.000

Fonte: Elaborado pelo Autor

3.3. Ferramentas Utilizadas

A linguagem de programação utilizada para desenvolver o modelo foi o Python, que possui uma série de bibliotecas disponíveis para computação científica e já otimizadas para precisão, tornando-a uma linguagem escalável para aplicações mais exigentes em termos computacionais. Há uma série de bibliotecas que já implementaram os algoritmos de aprendizagem de máquina profunda e já apresentam a capacidade de utilizar o poder computacional de placas de vídeo com a tecnologia Cuda.

3.3.1. Tensorflow

Tensorflow é uma biblioteca de código aberto focada em computação numérica. Sua arquitetura permite que as aplicações executem em mais de um processador ou em placas de vídeo com tecnologia CUDA da Nvidia. A biblioteca foi desenvolvida pelo "Google Brain Team" no departamento de Inteligência de máquina do Google para pesquisar aprendizado de máquina e redes neurais profundas.

3.3.2. Keras

Keras é uma biblioteca de alto nível para redes neurais capaz de realizar a interface com o Tensorflow e outras bibliotecas de aprendizado de máquina como CNTK e Theano. A biblioteca facilita a modelagem de rede neurais e a prototipagem, uma vez que já possui os algoritmos de otimização como RMSProp, Adam e *Mini Batch* já disponíveis para serem utilizados.

3.4. Arquitetura da Rede

Para cada um dos três modelos (modelo combinado, modelo remessa e modelo câmbio), foram testadas 80 arquiteturas de redes neurais com cross-validation com k =10 e 2 algoritmos de otimização de taxa de aprendizado α (Adam e RMSProp). Foram testadas arquiteturas de

rede entre 1 e 4 camadas ocultas com quantidades de neurônios por camada oculta variando entre 1 e 20 neurônios.

Optou-se por utilizar a função de ativação ReLu nos neurônios das camadas ocultas, que apresenta desempenho superior à função sigmoidal em termos de velocidade de treinamento. O neurônio de saída da rede neural, por sua vez, utiliza uma função de ativação softmax com quatro categorias correspondentes ao intervalo de valor financeiro médio por operação do cliente, como visto anteriormente na Tabela 8.

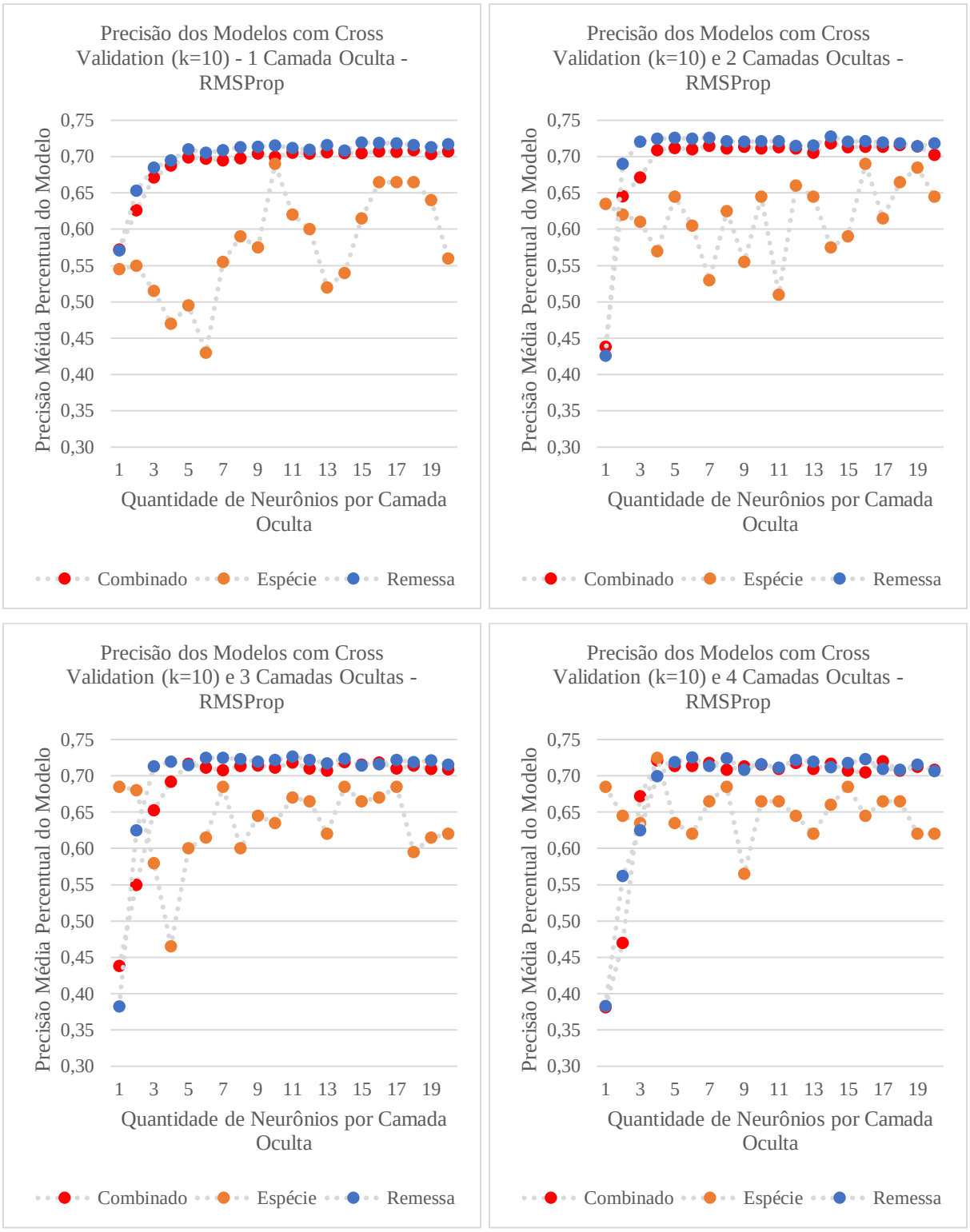
3.5. Resultado do Treinamento

O treinamento completo de todas as arquiteturas demorou cerca de 3 dias completos. Considerando a quantidade de dados que é pequena, apenas 60 mil clientes nos dados de treinamento, já é possível notar a importância das técnicas de otimização para o treinamento de redes neurais. Caso as técnicas de *mini-batch,* RMSProp e Adam não fossem utilizadas, o tempo necessário para obter esses mesmos resultados seriam maiores. Ao considerar modelos que requerem o processamento de milhões de dados ou tipos de dados mais pesados como imagens e texto, as técnicas de otimização tornam-se fundamentais para acelerar o processo de treinamento e obter bons resultados.

O método do *cross validation* com 10 iterações foi utilizado para avaliar a precisão das múltiplas arquiteturas testadas. O conjunto de treinamento correspondente à 80% dos dados do conjunto completo foi particionado aleatoriamente em 10, de modo que a cada iteração 9 partições eram utilizadas para treinar o modelo e 1 utilizada para avaliar a precisão. Ao final, calculou-se a média de cada uma das redes a partir das 10 avaliações e obteve-se o desvio padrão. O comparativo dos três modelos com os algoritmos de otimização RMSProp e Adam e cada uma das arquiteturas variando a quantidade de neurônios e camadas ocultas pode ser observado na Figura 14 e na Figura 15.

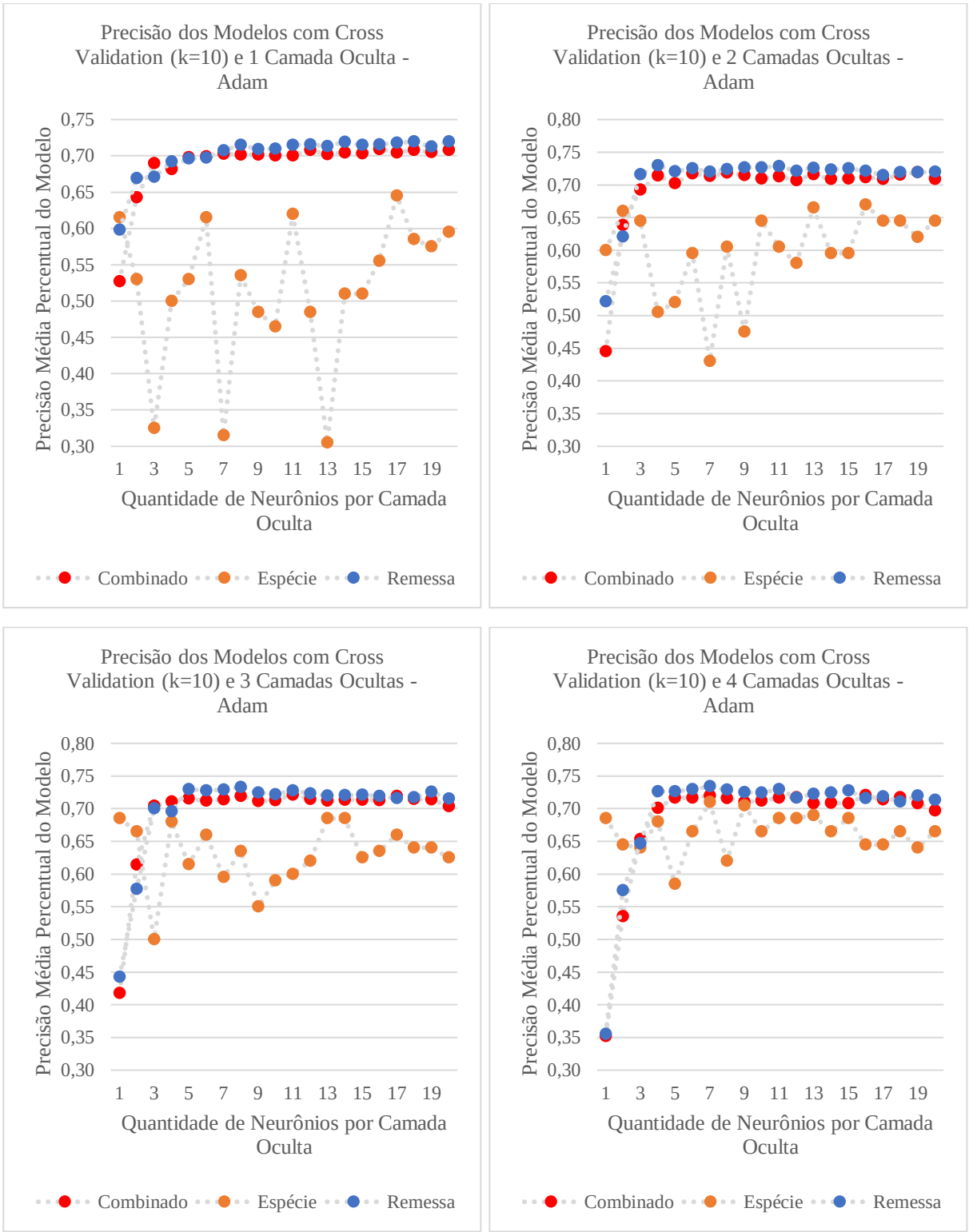
Os modelos combinado e da plataforma de remessa apresentaram uma menor variância ao longo das diversas arquiteturas testadas. É válido atentar que cada ponto no gráfico corresponde à média de 10 resultados testados em uma determinada arquitetura de rede com um algoritmo de otimização específico.

Figura 14 – Precisão dos Modelos com Algoritmo de Otimização RMSProp



Fonte: Elaborado pelo Autor

Figura 15 – Precisão dos Modelos com Algoritmo de Otimização Adam



Fonte: Elaborado pelo Autor

O modelo da plataforma de câmbio, por sua vez, não apresentou bons resultados. É possível observar que os resultados da precisão nas diversas arquiteturas não convergiram e apresentaram um desvio padrão elevado conforme a Tabela 9 e Tabela 10. Nota-se que em todas as arquiteturas não houve desvio padrão inferior a 0,10.

Ao comparar o modelo de câmbio com os outros modelos na Tabela 11, observa-se que, além dos outros modelos apresentarem resultados superiores de precisão, o desvio padrão dos modelos combinados e de remessa foram menores, indicando uma maior convergência dos resultados.

Uma possível explicação para os resultados ruins do modelo de câmbio pode ser a menor quantidade de operações nessa plataforma e a qualidade dos dados. Os clientes de câmbio costumam comprar esporadicamente, próximo à época de férias escolares, e não operam com tanta frequência como os clientes de remessa. A plataforma, por ser mais antiga, apresenta dados com menor qualidade que a da remessa, que possui dados mais completos sobre o cliente. O modelo combinado, por conseguir utilizar dados de ambas plataformas e cruzá-los para um mesmo cliente, permite uma maior precisão do modelo, mesmo quando considerando clientes de câmbio, pois muitos deles utilizam ambas plataformas.

Tabela 9 – Desvio Padrão da Precisão das Arquiteturas do Modelo Câmbio com Otimização Adam

Desvio Padrão da Precisão das Arquiteturas do Modelo Espécie com Otimização Adam				
Quantidade de Neurônios por Camada Oculta	Quantidade de Camadas Ocultas			
	1	**2**	**3**	**4**
1	0.242	0.201	0.161	0.161
2	0.223	0.126	0.158	0.217
3	0.264	0.253	0.310	0.197
4	0.256	0.218	0.131	0.194
5	0.157	0.302	0.253	0.230
6	0.249	0.261	0.169	0.158
7	0.234	0.287	0.135	0.150
8	0.250	0.246	0.205	0.154
9	0.238	0.262	0.283	0.162
10	0.205	0.177	0.145	0.158
11	0.190	0.222	0.190	0.161
12	0.226	0.218	0.178	0.161
13	0.231	0.203	0.161	0.207
14	0.203	0.135	0.161	0.158
15	0.145	0.208	0.191	0.161
16	0.211	0.172	0.130	0.177
17	0.137	0.177	0.169	0.177
18	0.263	0.177	0.118	0.158
19	0.216	0.178	0.197	0.162
20	0.175	0.177	0.191	0.158

Fonte: Elaborado pelo Autor

Tabela 10 – Desvio Padrão da Precisão das Arquiteturas do Modelo Câmbio com Otimização RMSProp

Desvio Padrão da Precisão das Arquiteturas do Modelo Espécie com Otimização RMSProp				
Quantidade de Neurônios por Camada Oculta	Quantidade de Camadas Ocultas			
	1	**2**	**3**	**4**
1	0.274	0.130	0.226	0.161
2	0.266	0.245	0.172	0.217
3	0.365	0.205	0.245	0.205
4	0.293	0.182	0.243	0.183
5	0.199	0.281	0.276	0.130
6	0.279	0.203	0.268	0.178
7	0.192	0.332	0.161	0.158
8	0.242	0.169	0.153	0.161
9	0.216	0.313	0.177	0.253
10	0.187	0.177	0.130	0.158
11	0.105	0.258	0.205	0.203
12	0.124	0.126	0.158	0.177
13	0.214	0.152	0.178	0.178
14	0.246	0.216	0.161	0.126
15	0.118	0.159	0.158	0.161
16	0.213	0.174	0.205	0.177
17	0.182	0.118	0.161	0.182
18	0.112	0.158	0.135	0.158
19	0.148	0.161	0.148	0.178
20	0.234	0.177	0.138	0.138

Fonte: Elaborado pelo Autor

Tabela 11 – Comparativo dos Modelos e Algoritmos de Otimização

Modelos	**Adam**		**RMSProp**	
	Precisão Média de Todas Arquiteturas	**Desvio Padrão**	**Precisão Média de Todas Arquiteturas**	**Desvio Padrão**
Combinado	69.00%	3.62%	68.80%	3.72%
Espécie	60.13%	19.52%	61.90%	19.16%
Remessa	70.04%	3.24%	69.74%	2.97%

Fonte: Elaborado pelo Autor

Para determinar se houve influência dos algoritmos de otimização sobre os resultados, foi realizado um teste de hipóteses para cada um dos modelos comparando a mesma arquitetura treinada com dois algoritmos de otimização da taxa de aprendizado distintos (RMSProp e Adam) para identificar se há uma diferença na média da precisão de cada arquitetura sendo treinada com técnicas diferentes.

Para o teste de hipóteses, temos a situação de dados não emparelhados. Os desvios padrões das duas populações são desconhecidos, mas podem ser assumidos como iguais, uma vez que se trata da comparação com mesmas arquiteturas. Desse modo, segundo Neto (2002), é necessário estimar o desvio padrão pela seguinte equação:

$$S_p^2 = \frac{(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2}{n_1 + n_2 - 2}$$

Nesse caso, S_1^2 e S_2^2 são as variâncias de cada uma das amostras obtidas no *cross-validation* com os dois algoritmos de otimização (RMSProp e Adam). Como o desvio σ é desconhecido, é necessário utilizar o t de student em conjunto com a estimativa S_p^2 com $n_1 + n_2 - 2$ graus de liberdade. Nesse caso, $n_1 = n_2 = k$ = 10. Desse modo, o teste poderá ser realizado pela seguinte equação:

$$t_{n_1+n_2-2} = \frac{(\overline{x_1} - \overline{x_2}) - \Delta}{S_p\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$

As seguintes hipóteses foram testadas:

$$\begin{cases} H_0: \mu_{RMSProp} = \mu_{Adam} \\ H_1: \mu_{RMSProp} \neq \mu_{Adam} \end{cases}$$

$$n_1 = n_2 = k = 10$$

$$\Delta = 0$$

$$\alpha = 5\% \rightarrow t_{18;\,5\%} = 1{,}734$$

A partir dos resultados, foi possível observar que um algoritmo superou o outro em apenas 3,75% das arquiteturas ao nível de significância de 5%. Nas outras arquiteturas, não há evidência de que um algoritmo apresentou melhores resultados que o outro nesse nível de significância. A Tabela 12 apresenta a quantidade de arquiteturas de redes neurais de cada um dos três modelos para cada resultado do teste de hipóteses comparando a média dos algoritmos de otimização Adam e RMSProp. Como é possível observar, das 80 combinações possíveis de camadas ocultas e quantidade de neurônios por camada (1 a 4 camadas ocultas e 1 a 20 neurônios por camada), a maior parte não apresentou diferença na média quando treinadas com algoritmos diferentes de taxa de aprendizado. O teste de hipóteses completo com os resultados para cada arquitetura pode ser encontrado no Apêndice A.

Tabela 12 – Quantidade de Arquiteturas por Resultado do Teste de Hipóteses

Modelo	$\mu_{Adam} > \mu_{RMSProp}$	$\mu_{RMSProp} > \mu_{Adam}$	$\mu_{RMSProp} = \mu_{Adam}$
Câmbio	1	4	75
Remessa	4	0	76
Combinado	0	0	80
Total	**5**	**4**	**231**

Fonte: Elaborado pelo Autor

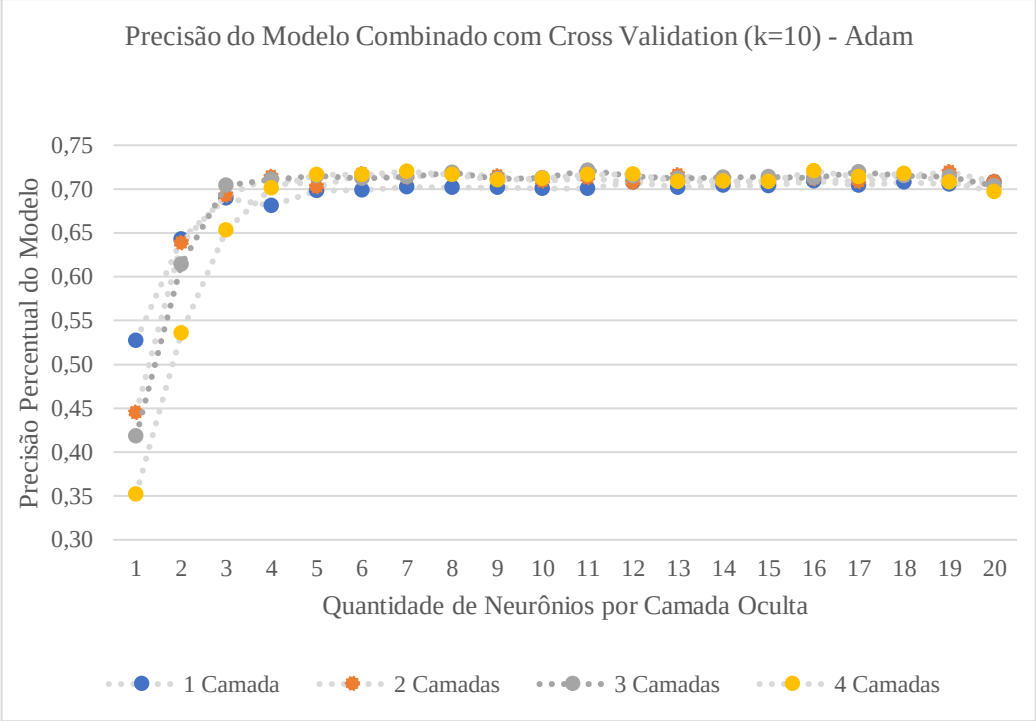
Por fim, escolheu-se o modelo mais adequado para ser implementado na base de dados. Para tanto, listou-se as maiores precisões dos modelos conforme a Figura 17 e optou-se pelo primeiro com 3 camadas ocultas e com 11 neurônios em cada uma delas. O modelo deverá ser treinado pelo algoritmo Adam ou RMSProp de otimização, pois, com base no teste de hipóteses, não há desconfiança de que um algoritmo seja superior ao outro nessa arquitetura de rede neural. A Figura 16 e Figura 17 apresentam a precisão do modelo combinado para as diferentes arquiteturas e algoritmos de otimização.

Tabela 13 – Dez melhores Precisões das Arquiteturas do Modelo Combinado

Camadas Ocultas	**Quantidade de Neurônios por Camada Oculta**	**Precisão**	**Desvio Padrão**	**Algoritmo de Otimização**
3 Camadas	11	72,15%	1,69%	Adam
4 Camadas	4	72,10%	2,73%	RMSProp
4 Camadas	16	72,05%	1,87%	Adam
4 Camadas	7	72,00%	1,84%	Adam
4 Camadas	17	72,00%	2,72%	RMSProp
3 Camadas	17	71,95%	2,45%	Adam
2 Camadas	19	71,93%	2,40%	Adam
3 Camadas	8	71,90%	2,28%	Adam
3 Camadas	14	71,90%	2,18%	RMSProp

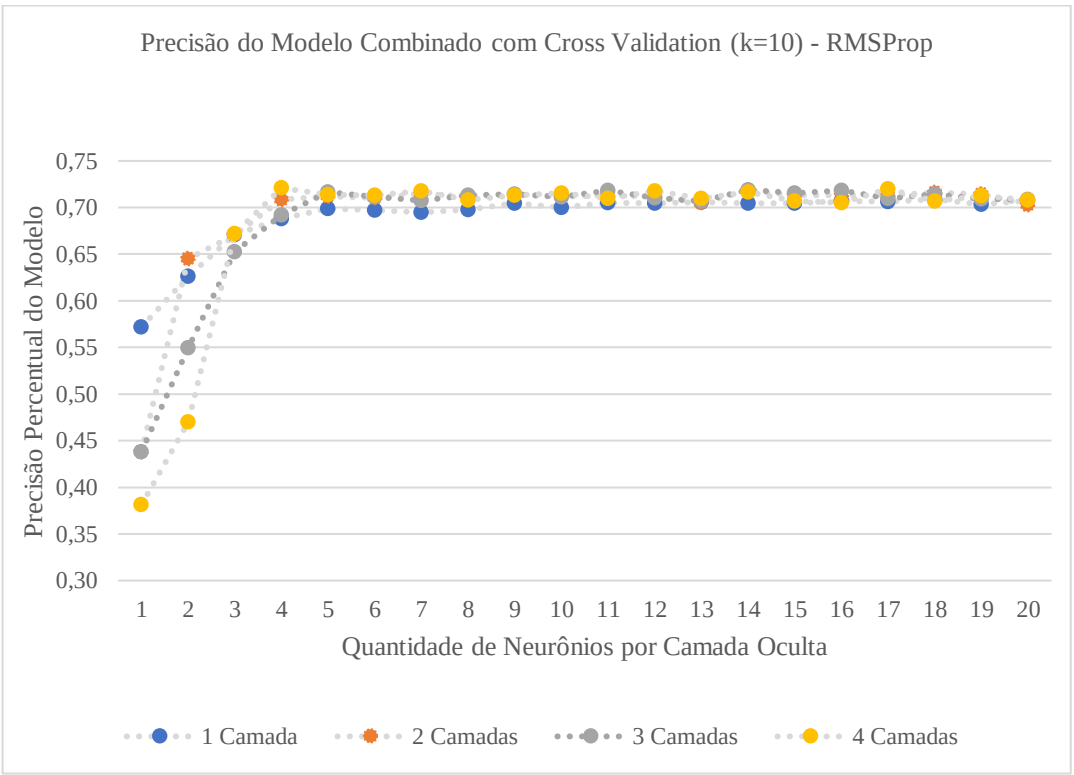
Fonte: Elaborado pelo Autor

Figura 16 – Precisão do Modelo Combinado com *Cross Validation* e Algoritmo de Otimização Adam



Fonte: Elaborado pelo Autor

Figura 17 - Precisão do Modelo Combinado com *Cross Validation* e Algoritmo de Otimização RMSProp



Fonte: Elaborado pelo Autor

A arquitetura da rede escolhida pode ser vista na Figura 18. A primeira camada é composta dos neurônios de entrada, sendo a quantidade de neurônios correspondente ao número de informações após as etapas de normalização. Há três camadas ocultas com função de ativação ReLu e 11 neurônios em cada camada. O neurônio de saída, por fim, possui a função de ativação *softmax*, que devolve as probabilidades de o cliente estar em uma das quatro categorias de ticket médio. A função *argmax* é aplicada pelo algoritmo para determinar qual a categoria mais provável do cliente com base nas probabilidades.

A variável n corresponde à quantidade de dados de entrada na rede neural e as variáveis $x_1 \dots x_n$ correspondem a cada um dos dados de entrada da rede neural que serão utilizados para determinar a categoria do cliente. No caso, como se trata do modelo combinado, há informações de entradas das plataformas online de remessa e câmbio. Todos os neurônios de uma camada estão conectados com os neurônios da camada seguinte.

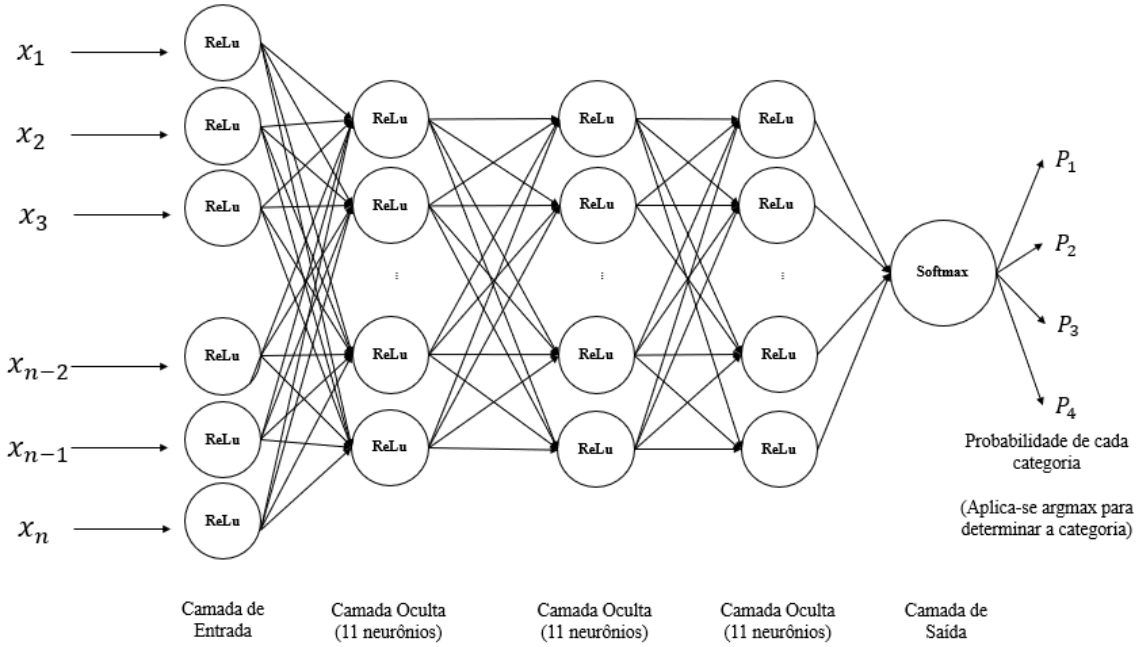
Por fim, definido o modelo, ele foi testado nos dados de validação, que nunca devem ser apresentados nas etapas de treinamento para não influenciar nos ajustes dos parâmetros. A Tabela 14 apresenta a precisão final do modelo escolhido nos dados de validação. Como é possível observar na precisão obtida de 70,30% no conjunto de validação, é possível concluir que o modelo apresentou bons resultados na generalização de dados nunca vistos antes pela rede neural.

Tabela 14 – Precisão Final do Modelo Escolhido

	Conjunto de Dados	
	Treinamento (80% dos dados)	Validação (20% dos dados)
Precisão	72,15%	70,30%

Fonte: Elaborado pelo Autor

Figura 18 – Rede Neural Escolhida



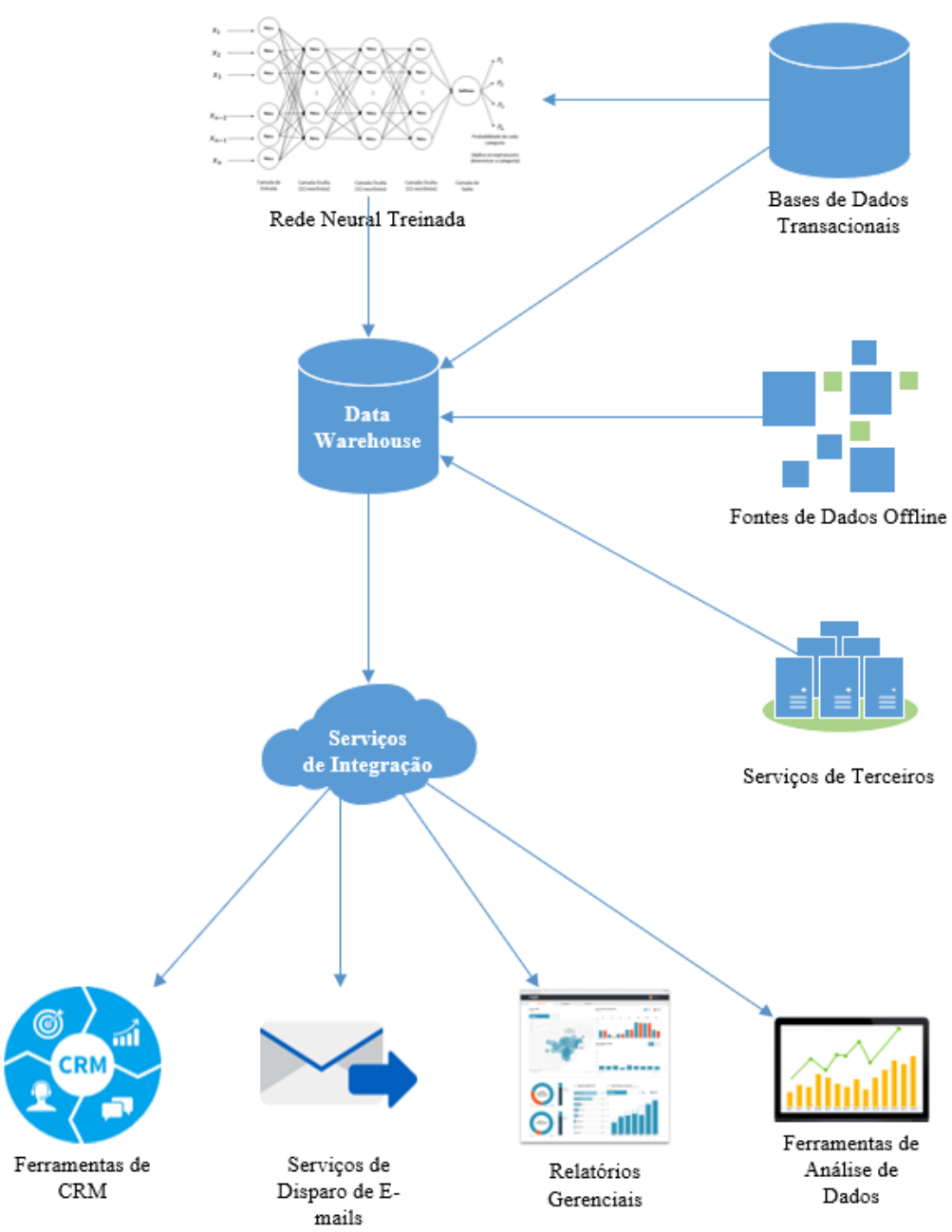
Fonte: Elaborado pelo Autor

3.6. Integração do Modelo com o Data Warehouse

Atualmente a empresa possui um *Data Warehouse*, que é um banco de dados com informações consolidadas e métricas prontas para serem utilizadas. Esse banco de dados é a fonte de informações para as ferramentas de CRM e disparos de e-mail, que serão as principais ferramentas a consumir a informação da classificação preditiva do cliente nas réguas de disparo de e-mails e de atendimento pela área comercial. Os dados presentes nesse banco de dados poderão ser utilizados em conjunto da categoria preditiva, de modo a permitir a formulação de estratégias de comunicação mais robustas. Algumas métricas interessantes que poderiam ser associadas a essa informação são a frequência de operação do cliente e a recência de operação, isto é, quanto tempo faz desde a última operação do cliente.

Nesse sentido, é fundamental que essa informação seja atualizada com frequência no banco de dados para os novos clientes que entrarem na plataforma e para os clientes existentes da base, uma vez que dependendo das mudanças das variáveis de entrada sua classificação poderá mudar. Essa atualização será feita por uma rotina que a cada determinado intervalo de tempo irá extrair as informações necessárias para classificar o cliente atualizar a base de dados com as informações preditivas. O esquema de integração completa esquematizado do fluxo dos dados pode ser observado na Figura 19.

Figura 19 – Arquitetura da Integração do Modelo com o *Data Warehouse*



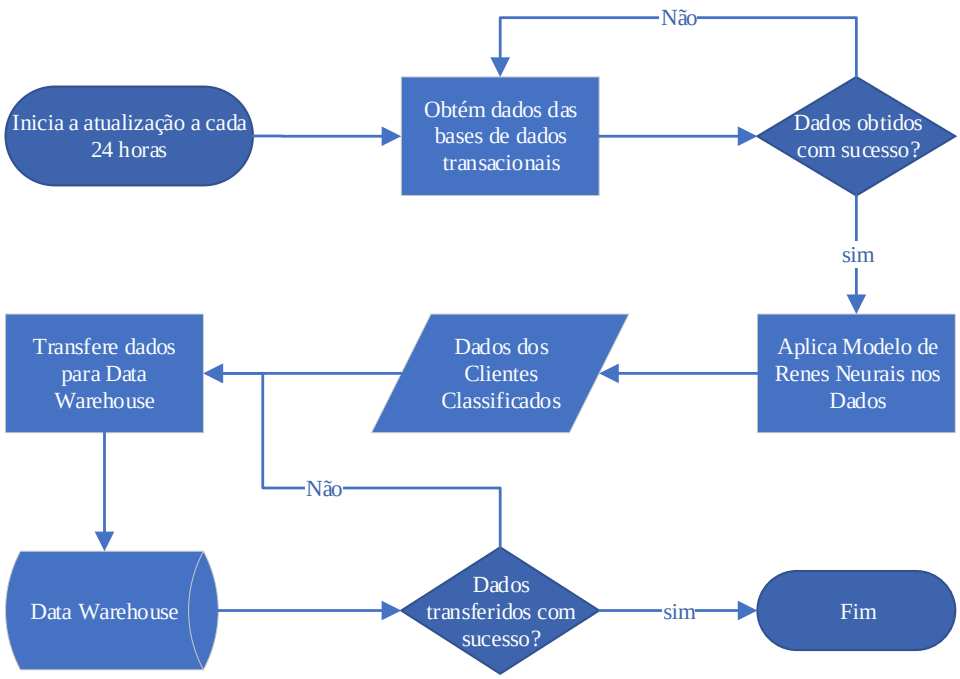
Fonte: Elaborado pelo Autor

O programa responsável por atualizar o banco de dados será enviado para um serviço na nuvem, que será executado de forma automática sem a necessidade de intervenção manual. Quando a base de dados do *Data Warehouse* estiver atualizada, um segundo programa realizará a atualização das informações dessa base de dados para as demais ferramentas, isto é, as ferramentas de marketing e CRM. De forma resumida, o programa que atualizará o ticket médio preditivo da base de clientes realizará as seguintes tarefas:

1. Extrair os dados das bases transacionais.
2. Aplicar o modelo de aprendizagem profunda já treinado nos dados para determinar o ticket médio preditivo do cliente.
3. Enviar os dados para o Data Warehouse.

O esquema de funcionamento do programa responsável por atualizar o ticket médio preditivo pode ser visto no diagrama da Figura 20.

Figura 20 – Funcionamento do Atualizador dos Dados pelo Modelo



Fonte: Elaborado pelo Autor

4. CONCLUSÃO

O presente trabalho buscou construir um método preditivo para a segmentação de clientes com base no potencial de receita média por operação para a empresa levando em consideração dados cadastrais e informações de visita nas plataformas de câmbio e remessas internacionais. Para o desenvolvimento do modelo para a resolução do problema, utilizou-se de técnicas de aprendizado de máquina, em específico, redes neurais de aprendizado profundo. A empresa não tinha processo definido para segmentar os clientes, de modo que a seleção manual de mais de 80 mil clientes é inviável.

Ao lidar com redes neurais, há uma série de hiperparâmetros que devem ser escolhidos e não necessariamente há recursos computacionais suficientes para otimizar todos os parâmetros. Para executar apenas os modelos testados nesse trabalho, por exemplo, foi necessário otimizar o código para utilizar a tecnologia CUDA de placas de vídeo da Nvidia em um computador Intel Core i7 com placa de vídeo GTX1050 Ti executando o modelo durante 3 dias consecutivos para treinar todas as redes neurais. Nesse contexto, surge a necessidade de se utilizar técnicas de otimização tanto para a taxa de aprendizado quanto para o método do gradiente.

O primeiro passo para possibilitar a segmentação de clientes em valor consistiu em identificar informações relevantes na base de dados, como geolocalização, idade, gênero, limite disponível para operar e tempo de acesso no site, por exemplo. Os dados foram tratados seguindo as metodologias apresentadas em Raschka e Mirjalili (2017), sendo necessário normalizar as variáveis numéricas ordinais e aplicar o método do *one-hot-encoding* nas variáveis nominais.

Foram testados três modelos com dados de entrada diferentes na rede neural para avaliar a influência da segmentação dos clientes em ambas plataformas. Nesse sentido, foram criados um modelo apenas da plataforma de remessas internacionais, um modelo apenas da plataforma de câmbio e um modelo combinado com os clientes de ambas plataformas, pois há clientes que operam em ambas plataformas.

Para cada um dos três modelos, foi necessário identificar qual deles apresentou melhor precisão. Para tanto, foram testadas diversas arquiteturas de redes variando a quantidade de camadas e a quantidade de neurônios em cada camada. A precisão foi avaliada com base no método de *cross-validation* com k = 10, conforme recomenda os estudos empíricos de Kohavi (1995). O método, que consiste em particionar os dados e utilizar cada uma das partições para validar os dados, permite avaliar a precisão de forma mais robusta que o método do Holdout,

também apresentado por Raschka e Mirjalili (2017). Outra vantagem do método do *cross-validaiton* é a redução da variância da média.

A função de ativação utilizada foi a ReLu, por ser considerada computacionalmente rápida e por reduzir os tempos de treinamento com uma convergência mais rápida conforme Patterson e Gibson (2017). A função de saída da rede, por sua vez, foi a *softmax*, a qual devolve uma distribuição de probabilidade para um conjunto de categorias mutuamente exclusivas, que, no caso, correspondem a cada uma das categorias de valor médio esperado por operação do cliente.

Um dos aspectos a se considerar no processo de treinamento de uma rede neural é o algoritmo de otimização dos pesos de cada neurônio, que permitem o modelo convergir de forma mais rápida. Dois algoritmos de otimização da taxa de aprendizagem foram testados em cada uma das configurações de rede neural, o Adam e o RMSProp. Para determinar se houve ou não melhor precisão de um algoritmo ou outro na rede neural, foram realizados testes de hipóteses ao nível de significância de 5% em cada uma das arquiteturas treinadas. Em relação à otimização do método do gradiente, foi utilizado o *mini-batch* com conjuntos de 50 clientes.

Com base nos resultados de treinamento das redes neurais, identificou-se que o modelo combinado era mais adequado, por apresentar uma precisão mais elevada e um menor desvio padrão do resultado no *cross-validation.* O modelo generalizou bem os clientes de ambas plataformas, apresentando um desempenho superior ao modelo individual para os clientes de câmbio.

Por fim, apresentou-se como o modelo com maior precisão será integrado na arquitetura atual de banco de dados para alimentar as plataformas de disparos de e-mail marketing e CRM. A implementação do valor preditivo do valor do cliente é fundamental para criar as réguas de relacionamento dos clientes e determinar quais são os clientes com maior potencial. Essa métrica nova será combinada com uma série de indicadores que estão sendo atualmente implementados no Data Warehouse para segmentar os clientes.

Dentre as melhorias futuras do modelo, é possível buscar novos tipos de informações para alimentar a rede neural e treinar novamente quando mais dados forem acumulados para melhorar o desempenho da rede. Também é possível treinar a rede com hiperparâmetros diferentes para encontrar uma arquitetura com maior desempenho. No entanto, é necessário um maior poder computacional para agilizar o treinamento, uma vez que demandará mais tempo de processamento.

REFERÊNCIAS BIBLIOGRÁFICA

BUDUMA, N. **Fundamentals of Deep Learning**.

GÉRON, A. **Hands-On Machine Learning with Scikit-Learn & TensorFlow**. O'Reilly Media, 2017.

KINGMA, D. P.; BA, J. L. Adam: a Method for Stochastic Optimization. **International Conference on Learning Representations 2015**, 2015.

KOHAVI, R. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In: Appears in the International Joint Conference on Articial Intelligence (IJCAI), **Anais**...1995.

MARIA BADEA, L. Predicting Consumer Behavior with Artificial Neural Networks. **Procedia Economics and Finance**, 2014.

PATTERSON, J.; GIBSON, A. **Deep Learning: A Practitioner's Approach**.

PEDRO LUIZ DE OLIVEIRA COSTA NETO. **Estatística**. 2. ed. São Paulo: Bluncher, 2002.

POLYAK, B. T. Some methods of speeding up the convergence of iteration methods. **USSR Computational Mathematics and Mathematical Physics**, 1964.

RASCHKA, S.; MIRJALILI, V. **Python Machine Learning**. 2. ed. Packt, 2017.

ROUTRAYA, G.; KANUNGO, P. Genetic algorithm based RNN structure for rayleigh fading MIMO channel estimation. In: Procedia Engineering, **Anais**...2012.

ROY, S. et al. Handwritten isolated Bangla compound character recognition: A new benchmark using a novel deep learning approach. **Pattern Recognition Letters**, 2017.

RUDER, S. An overview of gradient descent optimization algorithms∗. 2016.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **Nature**, 1986.

TSAIA, M.-J.; TAOA, Y.-H.; YUADIA, I. Deep Learning for Printed Document Source Identification. 2017.

XU, L. Representative splitting cross validation. **Chemometrics and Intelligent Laboratory Systems**, v. 183, 2018.

APÊNDICE A – RESULTADO DO TESTE DE HIPÓTESES COMPLETO

Modelo	Camadas Ocultas	Quantidade de Neurônios por Camada Oculta	Precisão RMSProp	S_{rms}	Precisão Adam	S_{Adam}	S_p^2	t	|t|	t_18;5%	Resultado	Melhor Modelo
Modelo Combinado	1	1	0,572	0,064	0,527	0,128	0,010	0,989	0,989	1,734	Aceito H0	
Modelo Combinado	1	2	0,626	0,043	0,643	0,056	0,002	-0,732	0,732	1,734	Aceito H0	
Modelo Combinado	1	3	0,671	0,032	0,690	0,033	0,001	-1,263	1,263	1,734	Aceito H0	
Modelo Combinado	1	4	0,688	0,036	0,681	0,023	0,001	0,488	0,488	1,734	Aceito H0	
Modelo Combinado	1	5	0,699	0,024	0,698	0,025	0,001	0,089	0,089	1,734	Aceito H0	
Modelo Combinado	1	6	0,697	0,033	0,699	0,028	0,001	-0,125	0,125	1,734	Aceito H0	
Modelo Combinado	1	7	0,695	0,026	0,702	0,025	0,001	-0,681	0,681	1,734	Aceito H0	
Modelo Combinado	1	8	0,698	0,029	0,701	0,027	0,001	-0,314	0,314	1,734	Aceito H0	
Modelo Combinado	1	9	0,704	0,029	0,701	0,024	0,001	0,245	0,245	1,734	Aceito H0	
Modelo Combinado	1	10	0,700	0,025	0,700	0,029	0,001	-0,041	0,041	1,734	Aceito H0	
Modelo Combinado	1	11	0,705	0,026	0,700	0,024	0,001	0,428	0,428	1,734	Aceito H0	
Modelo Combinado	1	12	0,704	0,030	0,708	0,024	0,001	-0,278	0,278	1,734	Aceito H0	
Modelo Combinado	1	13	0,706	0,025	0,702	0,024	0,001	0,351	0,351	1,734	Aceito H0	
Modelo Combinado	1	14	0,705	0,025	0,704	0,025	0,001	0,044	0,044	1,734	Aceito H0	
Modelo Combinado	1	15	0,705	0,027	0,703	0,023	0,001	0,107	0,107	1,734	Aceito H0	
Modelo Combinado	1	16	0,707	0,027	0,709	0,023	0,001	-0,197	0,197	1,734	Aceito H0	
Modelo Combinado	1	17	0,706	0,026	0,704	0,026	0,001	0,190	0,190	1,734	Aceito H0	
Modelo Combinado	1	18	0,709	0,030	0,708	0,023	0,001	0,062	0,062	1,734	Aceito H0	
Modelo Combinado	1	19	0,703	0,026	0,705	0,027	0,001	-0,144	0,144	1,734	Aceito H0	
Modelo Combinado	1	20	0,707	0,025	0,708	0,021	0,001	-0,023	0,023	1,734	Aceito H0	
Modelo Combinado	2	1	0,438	0,106	0,445	0,145	0,016	-0,119	0,119	1,734	Aceito H0	
Modelo Combinado	2	2	0,645	0,116	0,639	0,149	0,018	0,111	0,111	1,734	Aceito H0	
Modelo Combinado	2	3	0,671	0,125	0,693	0,054	0,009	-0,505	0,505	1,734	Aceito H0	
Modelo Combinado	2	4	0,709	0,029	0,714	0,028	0,001	-0,421	0,421	1,734	Aceito H0	
Modelo Combinado	2	5	0,712	0,026	0,702	0,041	0,001	0,612	0,612	1,734	Aceito H0	
Modelo Combinado	2	6	0,710	0,025	0,717	0,027	0,001	-0,592	0,592	1,734	Aceito H0	
Modelo Combinado	2	7	0,715	0,021	0,713	0,026	0,001	0,139	0,139	1,734	Aceito H0	
Modelo Combinado	2	8	0,711	0,031	0,718	0,024	0,001	-0,555	0,555	1,734	Aceito H0	
Modelo Combinado	2	9	0,714	0,025	0,714	0,022	0,001	-0,069	0,069	1,734	Aceito H0	
Modelo Combinado	2	10	0,711	0,026	0,709	0,018	0,000	0,198	0,198	1,734	Aceito H0	
Modelo Combinado	2	11	0,713	0,023	0,713	0,027	0,001	0,021	0,021	1,734	Aceito H0	
Modelo Combinado	2	12	0,711	0,024	0,707	0,023	0,001	0,395	0,395	1,734	Aceito H0	
Modelo Combinado	2	13	0,705	0,025	0,716	0,022	0,001	-0,986	0,986	1,734	Aceito H0	
Modelo Combinado	2	14	0,718	0,022	0,709	0,025	0,001	0,901	0,901	1,734	Aceito H0	
Modelo Combinado	2	15	0,713	0,027	0,710	0,027	0,001	0,303	0,303	1,734	Aceito H0	
Modelo Combinado	2	16	0,713	0,024	0,711	0,031	0,001	0,177	0,177	1,734	Aceito H0	
Modelo Combinado	2	17	0,713	0,024	0,709	0,027	0,001	0,425	0,425	1,734	Aceito H0	
Modelo Combinado	2	18	0,716	0,025	0,715	0,019	0,000	0,075	0,075	1,734	Aceito H0	
Modelo Combinado	2	19	0,714	0,021	0,719	0,024	0,001	-0,504	0,504	1,734	Aceito H0	
Modelo Combinado	2	20	0,703	0,019	0,709	0,027	0,001	-0,553	0,553	1,734	Aceito H0	

Modelo Combinado	3	1	0,438	0,125	0,418	0,138	0,017	0,341	0,341	1,734	Aceito H0	
Modelo Combinado	3	2	0,550	0,135	0,614	0,140	0,019	-1,046	1,046	1,734	Aceito H0	
Modelo Combinado	3	3	0,652	0,122	0,704	0,034	0,008	-1,289	1,289	1,734	Aceito H0	
Modelo Combinado	3	4	0,692	0,054	0,711	0,029	0,002	-0,971	0,971	1,734	Aceito H0	
Modelo Combinado	3	5	0,716	0,026	0,715	0,027	0,001	0,081	0,081	1,734	Aceito H0	
Modelo Combinado	3	6	0,711	0,023	0,712	0,032	0,001	-0,039	0,039	1,734	Aceito H0	
Modelo Combinado	3	7	0,708	0,027	0,714	0,022	0,001	-0,597	0,597	1,734	Aceito H0	
Modelo Combinado	3	8	0,713	0,029	0,719	0,023	0,001	-0,483	0,483	1,734	Aceito H0	
Modelo Combinado	3	9	0,714	0,025	0,711	0,022	0,001	0,254	0,254	1,734	Aceito H0	
Modelo Combinado	3	10	0,711	0,019	0,712	0,025	0,000	-0,121	0,121	1,734	Aceito H0	
Modelo Combinado	3	11	0,719	0,022	0,721	0,017	0,000	-0,331	0,331	1,734	Aceito H0	
Modelo Combinado	3	12	0,710	0,025	0,714	0,023	0,001	-0,389	0,389	1,734	Aceito H0	
Modelo Combinado	3	13	0,707	0,022	0,712	0,023	0,000	-0,536	0,536	1,734	Aceito H0	
Modelo Combinado	3	14	0,719	0,022	0,713	0,028	0,001	0,498	0,498	1,734	Aceito H0	
Modelo Combinado	3	15	0,715	0,021	0,714	0,023	0,000	0,173	0,173	1,734	Aceito H0	
Modelo Combinado	3	16	0,718	0,021	0,713	0,034	0,001	0,441	0,441	1,734	Aceito H0	
Modelo Combinado	3	17	0,710	0,026	0,719	0,024	0,001	-0,848	0,848	1,734	Aceito H0	
Modelo Combinado	3	18	0,714	0,020	0,715	0,029	0,001	-0,067	0,067	1,734	Aceito H0	
Modelo Combinado	3	19	0,709	0,031	0,714	0,021	0,001	-0,414	0,414	1,734	Aceito H0	
Modelo Combinado	3	20	0,709	0,021	0,704	0,029	0,001	0,450	0,450	1,734	Aceito H0	
Modelo Combinado	4	1	0,381	0,106	0,352	0,029	0,006	0,846	0,846	1,734	Aceito H0	
Modelo Combinado	4	2	0,470	0,135	0,535	0,189	0,027	-0,892	0,892	1,734	Aceito H0	
Modelo Combinado	4	3	0,672	0,108	0,653	0,102	0,011	0,403	0,403	1,734	Aceito H0	
Modelo Combinado	4	4	0,721	0,027	0,701	0,039	0,001	1,307	1,307	1,734	Aceito H0	
Modelo Combinado	4	5	0,714	0,026	0,716	0,027	0,001	-0,227	0,227	1,734	Aceito H0	
Modelo Combinado	4	6	0,713	0,027	0,716	0,021	0,001	-0,273	0,273	1,734	Aceito H0	
Modelo Combinado	4	7	0,718	0,026	0,720	0,018	0,000	-0,245	0,245	1,734	Aceito H0	
Modelo Combinado	4	8	0,708	0,025	0,716	0,027	0,001	-0,673	0,673	1,734	Aceito H0	
Modelo Combinado	4	9	0,713	0,022	0,710	0,026	0,001	0,274	0,274	1,734	Aceito H0	
Modelo Combinado	4	10	0,715	0,026	0,712	0,027	0,001	0,286	0,286	1,734	Aceito H0	
Modelo Combinado	4	11	0,710	0,025	0,716	0,022	0,001	-0,648	0,648	1,734	Aceito H0	
Modelo Combinado	4	12	0,718	0,021	0,717	0,025	0,001	0,047	0,047	1,734	Aceito H0	
Modelo Combinado	4	13	0,710	0,024	0,708	0,020	0,000	0,150	0,150	1,734	Aceito H0	
Modelo Combinado	4	14	0,716	0,026	0,709	0,022	0,001	0,701	0,701	1,734	Aceito H0	
Modelo Combinado	4	15	0,707	0,022	0,708	0,020	0,000	-0,128	0,128	1,734	Aceito H0	
Modelo Combinado	4	16	0,705	0,035	0,720	0,019	0,001	-1,236	1,236	1,734	Aceito H0	
Modelo Combinado	4	17	0,720	0,027	0,714	0,025	0,001	0,505	0,505	1,734	Aceito H0	
Modelo Combinado	4	18	0,707	0,027	0,717	0,026	0,001	-0,884	0,884	1,734	Aceito H0	
Modelo Combinado	4	19	0,713	0,027	0,708	0,023	0,001	0,413	0,413	1,734	Aceito H0	
Modelo Combinado	4	20	0,708	0,029	0,697	0,021	0,001	0,988	0,988	1,734	Aceito H0	
Modelo Câmbio	1	1	0,545	0,274	0,615	0,242	0,067	-0,605	0,605	1,734	Aceito H0	
Modelo Câmbio	1	2	0,550	0,266	0,530	0,223	0,060	0,182	0,182	1,734	Aceito H0	
Modelo Câmbio	1	3	0,515	0,365	0,325	0,264	0,101	1,335	1,335	1,734	Aceito H0	
Modelo Câmbio	1	4	0,470	0,293	0,500	0,256	0,076	-0,244	0,244	1,734	Aceito H0	
Modelo Câmbio	1	5	0,495	0,199	0,530	0,157	0,032	-0,436	0,436	1,734	Aceito H0	

Modelo Câmbio	1	6	0,430	0,279	0,615	0,249	0,070	-1,566	1,566	1,734	Aceito H0	
Modelo Câmbio	1	7	0,555	0,192	0,315	0,234	0,046	2,512	2,512	1,734	Rejetio H0	RMSProp
Modelo Câmbio	1	8	0,590	0,242	0,535	0,250	0,060	0,500	0,500	1,734	Aceito H0	
Modelo Câmbio	1	9	0,575	0,216	0,485	0,238	0,052	0,886	0,886	1,734	Aceito H0	
Modelo Câmbio	1	10	0,690	0,187	0,465	0,205	0,038	2,565	2,565	1,734	Rejetio H0	RMSProp
Modelo Câmbio	1	11	0,620	0,105	0,620	0,190	0,024	0,000	0,000	1,734	Aceito H0	
Modelo Câmbio	1	12	0,600	0,124	0,485	0,226	0,033	1,410	1,410	1,734	Aceito H0	
Modelo Câmbio	1	13	0,520	0,214	0,305	0,231	0,049	2,163	2,163	1,734	Rejetio H0	RMSProp
Modelo Câmbio	1	14	0,540	0,246	0,510	0,203	0,051	0,297	0,297	1,734	Aceito H0	
Modelo Câmbio	1	15	0,615	0,118	0,510	0,145	0,017	1,777	1,777	1,734	Rejetio H0	RMSProp
Modelo Câmbio	1	16	0,665	0,213	0,555	0,211	0,045	1,158	1,158	1,734	Aceito H0	
Modelo Câmbio	1	17	0,665	0,182	0,645	0,137	0,026	0,278	0,278	1,734	Aceito H0	
Modelo Câmbio	1	18	0,665	0,112	0,585	0,263	0,041	0,886	0,886	1,734	Aceito H0	
Modelo Câmbio	1	19	0,640	0,148	0,575	0,216	0,034	0,785	0,785	1,734	Aceito H0	
Modelo Câmbio	1	20	0,560	0,234	0,595	0,175	0,043	-0,378	0,378	1,734	Aceito H0	
Modelo Câmbio	2	1	0,635	0,130	0,600	0,201	0,029	0,461	0,461	1,734	Aceito H0	
Modelo Câmbio	2	2	0,620	0,245	0,660	0,126	0,038	-0,459	0,459	1,734	Aceito H0	
Modelo Câmbio	2	3	0,610	0,205	0,645	0,253	0,053	-0,340	0,340	1,734	Aceito H0	
Modelo Câmbio	2	4	0,570	0,182	0,505	0,218	0,040	0,723	0,723	1,734	Aceito H0	
Modelo Câmbio	2	5	0,645	0,281	0,520	0,302	0,085	0,958	0,958	1,734	Aceito H0	
Modelo Câmbio	2	6	0,605	0,203	0,595	0,261	0,055	0,096	0,096	1,734	Aceito H0	
Modelo Câmbio	2	7	0,530	0,332	0,430	0,287	0,096	0,721	0,721	1,734	Aceito H0	
Modelo Câmbio	2	8	0,625	0,169	0,605	0,246	0,045	0,212	0,212	1,734	Aceito H0	
Modelo Câmbio	2	9	0,555	0,313	0,475	0,262	0,083	0,619	0,619	1,734	Aceito H0	
Modelo Câmbio	2	10	0,645	0,177	0,645	0,177	0,031	0,000	0,000	1,734	Aceito H0	
Modelo Câmbio	2	11	0,510	0,258	0,605	0,222	0,058	-0,883	0,883	1,734	Aceito H0	
Modelo Câmbio	2	12	0,660	0,126	0,580	0,218	0,032	1,004	1,004	1,734	Aceito H0	
Modelo Câmbio	2	13	0,645	0,152	0,665	0,203	0,032	-0,250	0,250	1,734	Aceito H0	
Modelo Câmbio	2	14	0,575	0,216	0,595	0,135	0,032	-0,248	0,248	1,734	Aceito H0	
Modelo Câmbio	2	15	0,590	0,159	0,595	0,208	0,034	-0,060	0,060	1,734	Aceito H0	
Modelo Câmbio	2	16	0,690	0,174	0,670	0,172	0,030	0,258	0,258	1,734	Aceito H0	
Modelo Câmbio	2	17	0,615	0,118	0,645	0,177	0,023	-0,446	0,446	1,734	Aceito H0	
Modelo Câmbio	2	18	0,665	0,158	0,645	0,177	0,028	0,267	0,267	1,734	Aceito H0	
Modelo Câmbio	2	19	0,685	0,161	0,620	0,178	0,029	0,856	0,856	1,734	Aceito H0	
Modelo Câmbio	2	20	0,645	0,177	0,645	0,177	0,031	0,000	0,000	1,734	Aceito H0	
Modelo Câmbio	3	1	0,685	0,226	0,685	0,161	0,039	0,000	0,000	1,734	Aceito H0	
Modelo Câmbio	3	2	0,680	0,172	0,665	0,158	0,027	0,203	0,203	1,734	Aceito H0	
Modelo Câmbio	3	3	0,580	0,245	0,500	0,310	0,078	0,640	0,640	1,734	Aceito H0	
Modelo Câmbio	3	4	0,465	0,243	0,680	0,131	0,038	-2,464	2,464	1,734	Rejetio H0	Adam
Modelo Câmbio	3	5	0,600	0,276	0,615	0,253	0,070	-0,127	0,127	1,734	Aceito H0	
Modelo Câmbio	3	6	0,615	0,268	0,660	0,169	0,050	-0,449	0,449	1,734	Aceito H0	
Modelo Câmbio	3	7	0,685	0,161	0,595	0,135	0,022	1,353	1,353	1,734	Aceito H0	
Modelo Câmbio	3	8	0,600	0,153	0,635	0,205	0,033	-0,432	0,432	1,734	Aceito H0	
Modelo Câmbio	3	9	0,645	0,177	0,550	0,283	0,056	0,901	0,901	1,734	Aceito H0	
Modelo Câmbio	3	10	0,635	0,130	0,590	0,145	0,019	0,731	0,731	1,734	Aceito H0	

Modelo Câmbio	3	11	0,670	0,205	0,600	0,190	0,039	0,792	0,792	1,734	Aceito H0	
Modelo Câmbio	3	12	0,665	0,158	0,620	0,178	0,028	0,598	0,598	1,734	Aceito H0	
Modelo Câmbio	3	13	0,620	0,178	0,685	0,161	0,029	-0,856	0,856	1,734	Aceito H0	
Modelo Câmbio	3	14	0,685	0,161	0,685	0,161	0,026	0,000	0,000	1,734	Aceito H0	
Modelo Câmbio	3	15	0,665	0,158	0,625	0,191	0,031	0,509	0,509	1,734	Aceito H0	
Modelo Câmbio	3	16	0,670	0,205	0,635	0,130	0,030	0,455	0,455	1,734	Aceito H0	
Modelo Câmbio	3	17	0,685	0,161	0,660	0,169	0,027	0,339	0,339	1,734	Aceito H0	
Modelo Câmbio	3	18	0,595	0,135	0,640	0,118	0,016	-0,794	0,794	1,734	Aceito H0	
Modelo Câmbio	3	19	0,615	0,148	0,640	0,197	0,030	-0,320	0,320	1,734	Aceito H0	
Modelo Câmbio	3	20	0,620	0,138	0,625	0,191	0,028	-0,067	0,067	1,734	Aceito H0	
Modelo Câmbio	4	1	0,685	0,161	0,685	0,161	0,026	0,000	0,000	1,734	Aceito H0	
Modelo Câmbio	4	2	0,645	0,217	0,645	0,217	0,047	0,000	0,000	1,734	Aceito H0	
Modelo Câmbio	4	3	0,635	0,205	0,640	0,197	0,040	-0,056	0,056	1,734	Aceito H0	
Modelo Câmbio	4	4	0,725	0,183	0,680	0,194	0,036	0,533	0,533	1,734	Aceito H0	
Modelo Câmbio	4	5	0,635	0,130	0,585	0,230	0,035	0,597	0,597	1,734	Aceito H0	
Modelo Câmbio	4	6	0,620	0,178	0,665	0,158	0,028	-0,598	0,598	1,734	Aceito H0	
Modelo Câmbio	4	7	0,665	0,158	0,710	0,150	0,024	-0,653	0,653	1,734	Aceito H0	
Modelo Câmbio	4	8	0,685	0,161	0,620	0,154	0,025	0,923	0,923	1,734	Aceito H0	
Modelo Câmbio	4	9	0,565	0,253	0,705	0,162	0,045	-1,474	1,474	1,734	Aceito H0	
Modelo Câmbio	4	10	0,665	0,158	0,665	0,158	0,025	0,000	0,000	1,734	Aceito H0	
Modelo Câmbio	4	11	0,665	0,203	0,685	0,161	0,034	-0,244	0,244	1,734	Aceito H0	
Modelo Câmbio	4	12	0,645	0,177	0,685	0,161	0,029	-0,529	0,529	1,734	Aceito H0	
Modelo Câmbio	4	13	0,620	0,178	0,690	0,207	0,037	-0,811	0,811	1,734	Aceito H0	
Modelo Câmbio	4	14	0,660	0,126	0,665	0,158	0,020	-0,078	0,078	1,734	Aceito H0	
Modelo Câmbio	4	15	0,685	0,161	0,685	0,161	0,026	0,000	0,000	1,734	Aceito H0	
Modelo Câmbio	4	16	0,645	0,177	0,645	0,177	0,031	0,000	0,000	1,734	Aceito H0	
Modelo Câmbio	4	17	0,665	0,182	0,645	0,177	0,032	0,250	0,250	1,734	Aceito H0	
Modelo Câmbio	4	18	0,665	0,158	0,665	0,158	0,025	0,000	0,000	1,734	Aceito H0	
Modelo Câmbio	4	19	0,620	0,178	0,640	0,162	0,029	-0,263	0,263	1,734	Aceito H0	
Modelo Câmbio	4	20	0,620	0,138	0,665	0,158	0,022	-0,677	0,677	1,734	Aceito H0	
Modelo Remessa	1	1	0,571	0,066	0,598	0,052	0,004	-1,028	1,028	1,734	Aceito H0	
Modelo Remessa	1	2	0,653	0,033	0,669	0,029	0,001	-1,164	1,164	1,734	Aceito H0	
Modelo Remessa	1	3	0,685	0,019	0,671	0,033	0,001	1,130	1,130	1,734	Aceito H0	
Modelo Remessa	1	4	0,695	0,020	0,692	0,035	0,001	0,215	0,215	1,734	Aceito H0	
Modelo Remessa	1	5	0,710	0,022	0,696	0,024	0,001	1,363	1,363	1,734	Aceito H0	
Modelo Remessa	1	6	0,705	0,023	0,697	0,023	0,001	0,779	0,779	1,734	Aceito H0	
Modelo Remessa	1	7	0,709	0,018	0,708	0,019	0,000	0,152	0,152	1,734	Aceito H0	
Modelo Remessa	1	8	0,713	0,022	0,715	0,020	0,000	-0,208	0,208	1,734	Aceito H0	
Modelo Remessa	1	9	0,713	0,019	0,709	0,017	0,000	0,528	0,528	1,734	Aceito H0	
Modelo Remessa	1	10	0,715	0,026	0,710	0,021	0,001	0,536	0,536	1,734	Aceito H0	
Modelo Remessa	1	11	0,712	0,020	0,715	0,019	0,000	-0,373	0,373	1,734	Aceito H0	
Modelo Remessa	1	12	0,709	0,016	0,716	0,019	0,000	-0,782	0,782	1,734	Aceito H0	
Modelo Remessa	1	13	0,716	0,023	0,713	0,018	0,000	0,319	0,319	1,734	Aceito H0	
Modelo Remessa	1	14	0,708	0,020	0,719	0,017	0,000	-1,316	1,316	1,734	Aceito H0	
Modelo Remessa	1	15	0,719	0,017	0,715	0,021	0,000	0,548	0,548	1,734	Aceito H0	

Modelo Remessa	1	16	0,719	0,020	0,715	0,023	0,000	0,362	0,362	1,734	Aceito H0	
Modelo Remessa	1	17	0,718	0,015	0,718	0,019	0,000	0,033	0,033	1,734	Aceito H0	
Modelo Remessa	1	18	0,716	0,021	0,719	0,028	0,001	-0,311	0,311	1,734	Aceito H0	
Modelo Remessa	1	19	0,713	0,023	0,712	0,021	0,000	0,050	0,050	1,734	Aceito H0	
Modelo Remessa	1	20	0,717	0,019	0,719	0,022	0,000	-0,219	0,219	1,734	Aceito H0	
Modelo Remessa	2	1	0,426	0,117	0,522	0,165	0,020	-1,499	1,499	1,734	Aceito H0	
Modelo Remessa	2	2	0,690	0,041	0,621	0,139	0,010	1,514	1,514	1,734	Aceito H0	
Modelo Remessa	2	3	0,721	0,023	0,716	0,013	0,000	0,562	0,562	1,734	Aceito H0	
Modelo Remessa	2	4	0,725	0,017	0,730	0,017	0,000	-0,648	0,648	1,734	Aceito H0	
Modelo Remessa	2	5	0,726	0,022	0,720	0,016	0,000	0,656	0,656	1,734	Aceito H0	
Modelo Remessa	2	6	0,725	0,026	0,725	0,020	0,001	-0,024	0,024	1,734	Aceito H0	
Modelo Remessa	2	7	0,726	0,019	0,720	0,019	0,000	0,658	0,658	1,734	Aceito H0	
Modelo Remessa	2	8	0,721	0,022	0,724	0,026	0,001	-0,249	0,249	1,734	Aceito H0	
Modelo Remessa	2	9	0,721	0,019	0,726	0,023	0,000	-0,630	0,630	1,734	Aceito H0	
Modelo Remessa	2	10	0,721	0,021	0,726	0,015	0,000	-0,633	0,633	1,734	Aceito H0	
Modelo Remessa	2	11	0,721	0,023	0,728	0,018	0,000	-0,767	0,767	1,734	Aceito H0	
Modelo Remessa	2	12	0,715	0,020	0,721	0,023	0,000	-0,673	0,673	1,734	Aceito H0	
Modelo Remessa	2	13	0,716	0,023	0,726	0,030	0,001	-0,868	0,868	1,734	Aceito H0	
Modelo Remessa	2	14	0,728	0,021	0,723	0,020	0,000	0,540	0,540	1,734	Aceito H0	
Modelo Remessa	2	15	0,721	0,017	0,725	0,024	0,000	-0,500	0,500	1,734	Aceito H0	
Modelo Remessa	2	16	0,721	0,025	0,721	0,017	0,000	-0,025	0,025	1,734	Aceito H0	
Modelo Remessa	2	17	0,719	0,025	0,714	0,024	0,001	0,453	0,453	1,734	Aceito H0	
Modelo Remessa	2	18	0,718	0,023	0,719	0,021	0,000	-0,075	0,075	1,734	Aceito H0	
Modelo Remessa	2	19	0,714	0,020	0,720	0,013	0,000	-0,722	0,722	1,734	Aceito H0	
Modelo Remessa	2	20	0,718	0,023	0,720	0,021	0,000	-0,153	0,153	1,734	Aceito H0	
Modelo Remessa	3	1	0,382	0,074	0,442	0,131	0,011	-1,260	1,260	1,734	Aceito H0	
Modelo Remessa	3	2	0,625	0,092	0,577	0,138	0,014	0,909	0,909	1,734	Aceito H0	
Modelo Remessa	3	3	0,713	0,035	0,701	0,044	0,002	0,685	0,685	1,734	Aceito H0	
Modelo Remessa	3	4	0,720	0,018	0,696	0,106	0,006	0,704	0,704	1,734	Aceito H0	
Modelo Remessa	3	5	0,715	0,020	0,730	0,020	0,000	-1,661	1,661	1,734	Aceito H0	
Modelo Remessa	3	6	0,724	0,026	0,728	0,028	0,001	-0,261	0,261	1,734	Aceito H0	
Modelo Remessa	3	7	0,725	0,033	0,729	0,023	0,001	-0,332	0,332	1,734	Aceito H0	
Modelo Remessa	3	8	0,723	0,024	0,733	0,017	0,000	-1,072	1,072	1,734	Aceito H0	
Modelo Remessa	3	9	0,719	0,020	0,725	0,023	0,000	-0,560	0,560	1,734	Aceito H0	
Modelo Remessa	3	10	0,722	0,020	0,722	0,023	0,000	0,000	0,000	1,734	Aceito H0	
Modelo Remessa	3	11	0,726	0,029	0,727	0,022	0,001	-0,086	0,086	1,734	Aceito H0	
Modelo Remessa	3	12	0,722	0,018	0,723	0,022	0,000	-0,189	0,189	1,734	Aceito H0	
Modelo Remessa	3	13	0,717	0,024	0,720	0,018	0,000	-0,234	0,234	1,734	Aceito H0	
Modelo Remessa	3	14	0,724	0,021	0,720	0,026	0,001	0,326	0,326	1,734	Aceito H0	
Modelo Remessa	3	15	0,714	0,014	0,721	0,020	0,000	-0,892	0,892	1,734	Aceito H0	
Modelo Remessa	3	16	0,716	0,018	0,719	0,020	0,000	-0,367	0,367	1,734	Aceito H0	
Modelo Remessa	3	17	0,722	0,019	0,716	0,020	0,000	0,669	0,669	1,734	Aceito H0	
Modelo Remessa	3	18	0,719	0,017	0,717	0,016	0,000	0,165	0,165	1,734	Aceito H0	
Modelo Remessa	3	19	0,721	0,026	0,726	0,018	0,001	-0,461	0,461	1,734	Aceito H0	
Modelo Remessa	3	20	0,716	0,016	0,715	0,024	0,000	0,027	0,027	1,734	Aceito H0	

Modelo Remessa	4	1	0,383	0,087	0,355	0,023	0,004	0,985	0,985	1,734	Aceito H0	
Modelo Remessa	4	2	0,562	0,178	0,575	0,157	0,028	-0,171	0,171	1,734	Aceito H0	
Modelo Remessa	4	3	0,625	0,139	0,647	0,149	0,021	-0,341	0,341	1,734	Aceito H0	
Modelo Remessa	4	4	0,700	0,033	0,726	0,026	0,001	-2,029	2,029	1,734	Rejetio H0	Adam
Modelo Remessa	4	5	0,719	0,026	0,726	0,019	0,001	-0,747	0,747	1,734	Aceito H0	
Modelo Remessa	4	6	0,725	0,025	0,730	0,023	0,001	-0,394	0,394	1,734	Aceito H0	
Modelo Remessa	4	7	0,713	0,022	0,734	0,021	0,000	-2,166	2,166	1,734	Rejetio H0	Adam
Modelo Remessa	4	8	0,724	0,019	0,729	0,024	0,000	-0,498	0,498	1,734	Aceito H0	
Modelo Remessa	4	9	0,708	0,017	0,725	0,023	0,000	-1,872	1,872	1,734	Rejetio H0	Adam
Modelo Remessa	4	10	0,716	0,022	0,725	0,026	0,001	-0,825	0,825	1,734	Aceito H0	
Modelo Remessa	4	11	0,711	0,024	0,730	0,020	0,000	-1,887	1,887	1,734	Rejetio H0	Adam
Modelo Remessa	4	12	0,722	0,024	0,717	0,025	0,001	0,499	0,499	1,734	Aceito H0	
Modelo Remessa	4	13	0,719	0,022	0,723	0,020	0,000	-0,366	0,366	1,734	Aceito H0	
Modelo Remessa	4	14	0,712	0,025	0,724	0,016	0,000	-1,317	1,317	1,734	Aceito H0	
Modelo Remessa	4	15	0,718	0,023	0,728	0,015	0,000	-1,146	1,146	1,734	Aceito H0	
Modelo Remessa	4	16	0,723	0,020	0,716	0,016	0,000	0,840	0,840	1,734	Aceito H0	
Modelo Remessa	4	17	0,709	0,024	0,719	0,025	0,001	-0,860	0,860	1,734	Aceito H0	
Modelo Remessa	4	18	0,708	0,016	0,710	0,019	0,000	-0,277	0,277	1,734	Aceito H0	
Modelo Remessa	4	19	0,716	0,026	0,720	0,019	0,001	-0,413	0,413	1,734	Aceito H0	
Modelo Remessa	4	20	0,707	0,036	0,713	0,019	0,001	-0,499	0,499	1,734	Aceito H0	

APÊNDICE B – MODELAGEM DO PROBLEMA EM PYTHON

```
"""
R -> só remessa
E -> só especie
C -> combinado
"""
identifier = uuid.uuid4().hex
MODO = 'C'
kfold_splits = 10

from_cache = True
if from_cache == False:
    with BeetechConnection() as conn:
        df = pd.read_sql(queries.ticket_medio_lat(), conn)

    with PiwikConnection(use_tunnel=False) as conn:
        visit_df = pd.read_sql(queries.email_visit_data(), conn)
    df.to_pickle("cache/{}_data.pckl".format(MODO))
    visit_df.to_pickle("cache/{}_visit_df.pckl".format(MODO))
else:
    df = pd.read_pickle("cache/{}_data.pckl".format(MODO))
    visit_df = pd.read_pickle("cache/{}_visit_df.pckl".format(MODO))

if MODO == 'R':
    df = df[df['y_rem'] != -1]
elif MODO == 'E':
    df = df[df['y_esp'] != -1]

final_df = pd.merge(df, visit_df, how='left', left_on='email',
right_on='user_id')

normalize_cols =
['limit_rem','ops_remessa','ops_especie','prim_tickt_esp','prim_tickt_rem',
'lim_aprovado_requerimento','r_limite_disponivel','r_limite_aprovado',
                  'visitor_count_visits_remessa',
'visit_total_actions_remessa', 'visit_total_interactions_remessa',
                  'visit_total_time_remessa',
'visitor_count_visits_especie', 'visit_total_actions_especie',
                  'visit_total_interactions_especie',
'visit_total_time_especie']


binary = ['windows_remessa', 'ios_remessa', 'android_remessa',
'mac_remessa', 'windows_especie', 'ios_especie', 'android_especie',
'mac_especie']
remessa_columns = ['visitor_count_visits_remessa',
'visit_total_actions_remessa', 'visit_total_interactions_remessa',
'visit_total_time_remessa', 'windows_remessa', 'ios_remessa',
'android_remessa', 'mac_remessa', 'lim_aprovado_requerimento',
'prim_tickt_rem', 'has_partner', 'limit_rem', 'r_limite_disponivel',
'r_limite_aprovado']
especie_columns = ['visitor_count_visits_especie',
'visit_total_actions_especie', 'visit_total_interactions_especie',
```

```
'visit_total_time_especie', 'windows_especie', 'ios_especie',
'android_especie', 'mac_especie', 'prim_tickt_esp', 'name']

df = final_df.drop(['id', 'email','user_id'],axis=1)

for col in binary:
    df[col] = df[col].fillna(0)

for col in normalize_cols:
    if col in df:
        df[col] = (df[col]-df[col].mean())/df[col].std()
        df[col] = df[col].fillna(df[col].mean())

if MODO == 'R':
    df = df.drop(especie_columns, axis=1)
elif MODO == 'E':
    df = df.drop(remessa_columns, axis=1)

df['x_coord'] = np.cos(df['lat']) * np.cos(df['lng'])
df['y_coord'] = np.cos(df['lat']) * np.sin(df['lng'])
df['z_coord'] = np.sin(df['lat'])
df = pd.get_dummies(df)

to_drop = ['y','y_rem','y_esp', 'lng', 'lat']

X = df.drop(to_drop, axis=1)

if MODO == 'R':
    y = df[['y_rem']]
elif MODO == 'E':
    y = df[['y_esp']]
elif MODO == 'C':
    y = df[['y']]



batch = 50
epochs = 50

kfold = KFold(n_splits=kfold_splits, shuffle=True, random_state=42)
res = []
for j in range(0,4):
    camadas = j+1
    for i in range(1,21):
        cvscores = []
        cvloss = []

        for train, test in kfold.split(X, y):
            kf_X_train = X.iloc[train]
            kf_X_val = X.iloc[test]
            kf_y_train = y.iloc[train]
            kf_y_val = y.iloc[test]
```

```
            kf_y_train = keras.utils.to_categorical(kf_y_train,
num_classes=4)
            kf_y_val = keras.utils.to_categorical(kf_y_val, num_classes=4)

            model = Sequential()
            n_features = X.shape[1]
            model.add(Dense(i, activation='relu', input_dim=n_features))
            for k in range(j):
                model.add(Dense(i, activation='relu'))

            model.add(Dense(4, activation='softmax'))

            model.compile(optimizer='adam',
                          loss='categorical_crossentropy',
                          metrics=['accuracy'])

            model.fit(kf_X_train, kf_y_train, epochs=epochs,
batch_size=batch, verbose=0)
            score_test = model.evaluate(kf_X_val, kf_y_val,
batch_size=batch)
            cvscores.append(score_test[1])
            cvloss.append(score_test[0])

        res.append([MODO, camadas, batch, epochs, i, kfold_splits,
np.mean(cvscores), np.std(cvscores), np.mean(cvloss), np.std(cvloss)])
        df = pd.DataFrame(res)
        df.columns = ['modo', 'camadas ocultas', 'batch', 'epochs',
'quantidade de neurônios', 'k_fold_cv', 'cv_acc',
                      'cv_acc_std', 'cv_loss', 'cv_loss_std']
        df.to_excel('result/kfold_adam/10_fold_cv_{}_{}.xlsx'.format(MODO,
identifier), index=False)
```

APÊNDICE C – CONSULTAS SQL PARA EXTRAÇÃO DOS DADOS

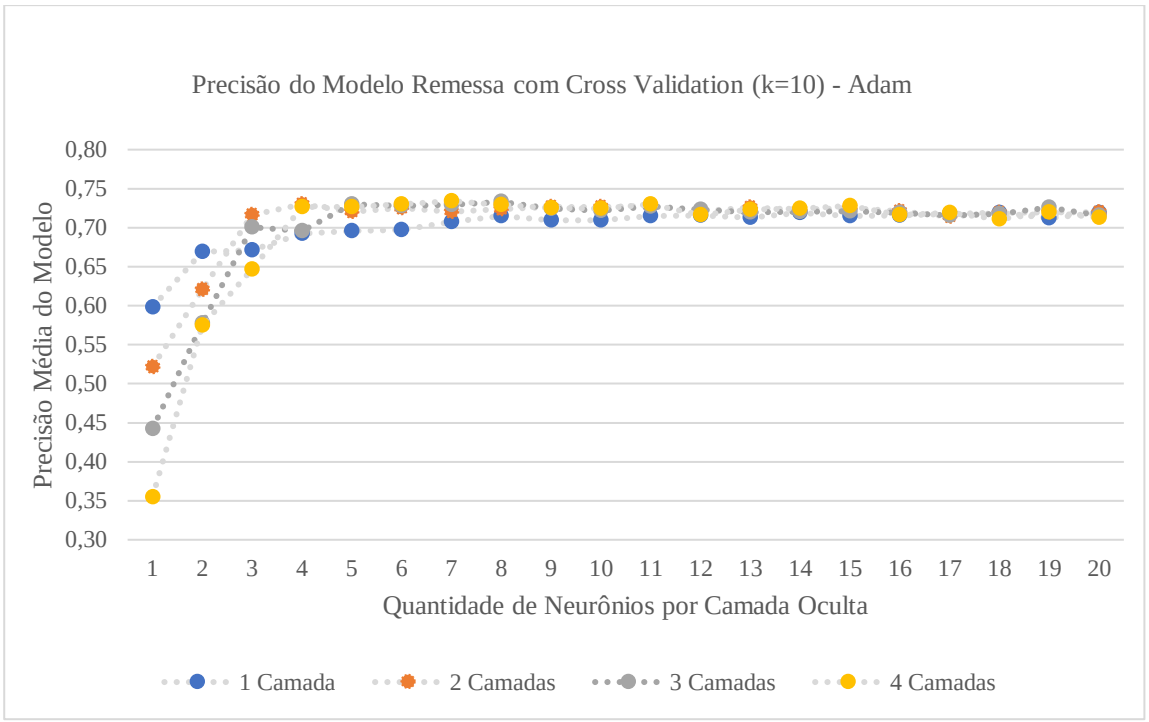
```
def ticket_medio_lat():
    return """
WITH ticket_medio as (
    SELECT id_customer, SUM(gmv)/SUM(ops) as ticket_medio,
    SUM(CASE WHEN plataforma = 'R' THEN gmv ELSE 0 END)/NULLIF(SUM(CASE
WHEN plataforma = 'R' THEN ops ELSE 0 END),0) as ticket_medio_remessa,
    SUM(CASE WHEN plataforma = 'E' THEN gmv ELSE 0 END)/NULLIF(SUM(CASE
WHEN plataforma = 'E' THEN ops ELSE 0 END),0) as ticket_medio_especie,
    SUM(CASE WHEN plataforma = 'R' THEN ops ELSE 0 END) as ops_remessa,
    SUM(CASE WHEN plataforma = 'E' THEN ops ELSE 0 END) as ops_especie
    FROM (
      SELECT id_customer, sum(total_value) as gmv, COUNT(*) ops, 'R' as
plataforma FROM tbl_remittance_operation o WHERE o.id_status = 10 AND
o.created_at < '20180101'
      GROUP BY 1
      UNION
      SELECT id_customer, sum(total_value) as gmv, COUNT(*) ops, 'E' as
plataforma FROM tbl_operation o WHERE o.id_status in (14,6,7) AND
o.created_at < '20180101'
      GROUP BY 1
    ) as x GROUP BY 1
)
,
region as (
    SELECT id_customer, state, city, geoposition FROM (
    SELECT id_customer,state, city, geoposition, RANK() OVER (PARTITION BY
id_customer ORDER BY created_at) FROM tbl_customer_address
    ) as x WHERE rank = 1
),
first_op_rem as (
SELECT * FROM (
    SELECT o.id_customer, o.total_value, RANK() OVER (PARTITION BY
id_customer ORDER BY created_at) FROM tbl_remittance_operation o WHERE
    o.id_status in (7,8,9,10,13) AND o.created_at < '{end_date}'
    ) as sub WHERE rank = 1
),
first_op_esp AS (
    SELECT * FROM (
    SELECT o.id_customer, o.total_value, RANK() OVER (PARTITION BY
id_customer ORDER BY created_at) FROM tbl_operation o WHERE
    o.id_status in (9, 14) AND o.created_at < '{end_date}'
    ) as sub WHERE rank = 1
 ),
lim AS (SELECT id_customer, SUM(limit_approved) as
lim_aprovado_requerimento FROM beecambio.tbl_requirement WHERE
id_requirement_status = 3
  GROUP BY 1
),
limites AS (
select (credit+initial)-(debit-refund) r_limite_disponivel,
       (((debit-refund) / (credit+initial)) * 100) as r_limite_utilizado,
       (credit+initial) as r_limite_aprovado,
```

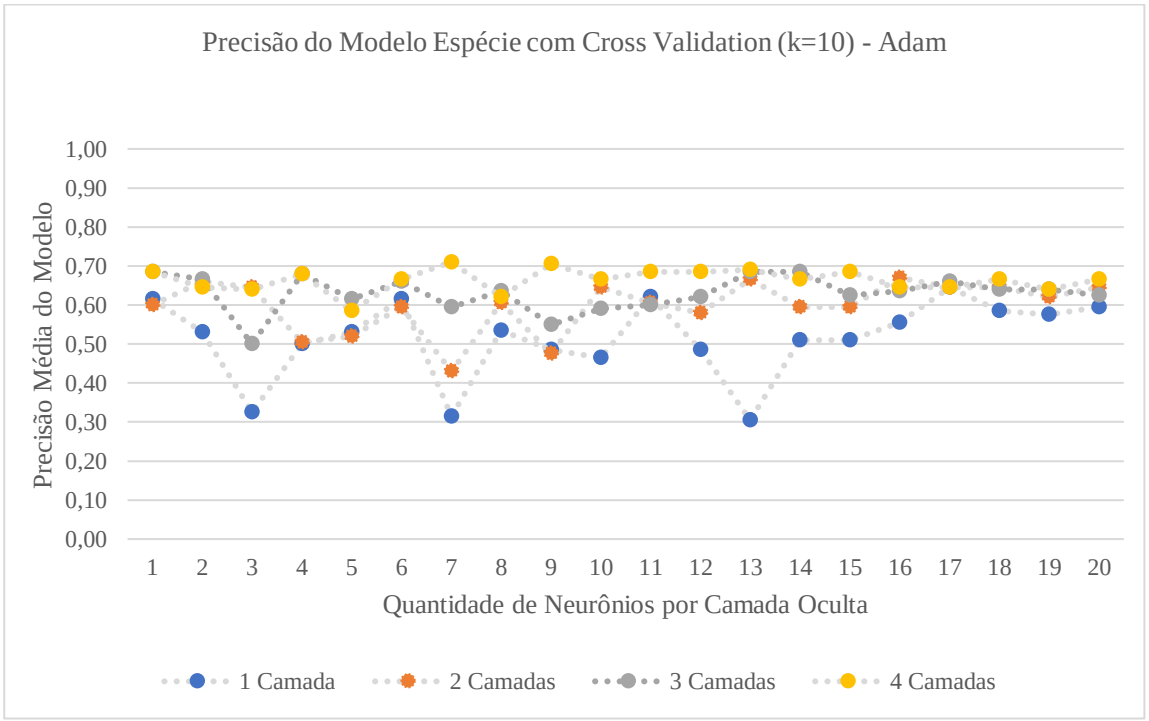
```
        id_customer
        from ( select id_customer,
                  sum(case when transaction_type in ('D') then value else 0
end) as DEBIT,
                  sum(case when transaction_type in ('R') then value else 0
end) as REFUND,
                  sum(case when transaction_type in ('C') then value else 0
end) as CREDIT,
                  sum(case when transaction_type in ('I') then value else 0
end) as INITIAL
          FROM tbl_limit_history WHERE deleted = false GROUP BY id_customer)
as total)
SELECT
c.email,
CASE WHEN ticket_medio.ticket_medio < 4000  THEN 0
WHEN ticket_medio.ticket_medio < 7000       THEN 1
WHEN ticket_medio.ticket_medio < 10000      THEN 2
                                          ELSE 3
END as y,

CASE
WHEN ticket_medio.ticket_medio_remessa is NULL THEN -1
WHEN ticket_medio.ticket_medio_remessa < 4000  THEN 0
WHEN ticket_medio.ticket_medio_remessa < 7000       THEN 1
WHEN ticket_medio.ticket_medio_remessa < 10000      THEN 2
                                           ELSE 3
END as y_rem,

CASE
WHEN ticket_medio.ticket_medio_especie is NULL THEN -1
WHEN ticket_medio.ticket_medio_especie < 4000  THEN 0
WHEN ticket_medio.ticket_medio_especie < 7000       THEN 1
WHEN ticket_medio.ticket_medio_especie < 10000      THEN 2
                                           ELSE 3
END as y_esp,
lim.lim_aprovado_requerimento,

first_op_esp.total_value as prim_tickt_esp,
first_op_rem.total_value as prim_tickt_rem,
CASE WHEN id_office is not NULL THEN 1 ELSE 0 END as has_partner,
c.id,
(current_date - birthday)::FLOAT/(100*365) as dias_nascimento,
CASE WHEN remittance_limit < 0 THEN 0 ELSE remittance_limit END as
limit_rem,
CASE WHEN genre = 'F' THEN 1 ELSE 0 END as female,
CASE WHEN genre = 'M' THEN 1 ELSE 0 END as male,
COALESCE(region.geoposition[0],0) as lng,
COALESCE(region.geoposition[1],0) as lat,
p.name,
limites.r_limite_disponivel,
limites.r_limite_aprovado
FROM beecambio.tbl_customer c
```

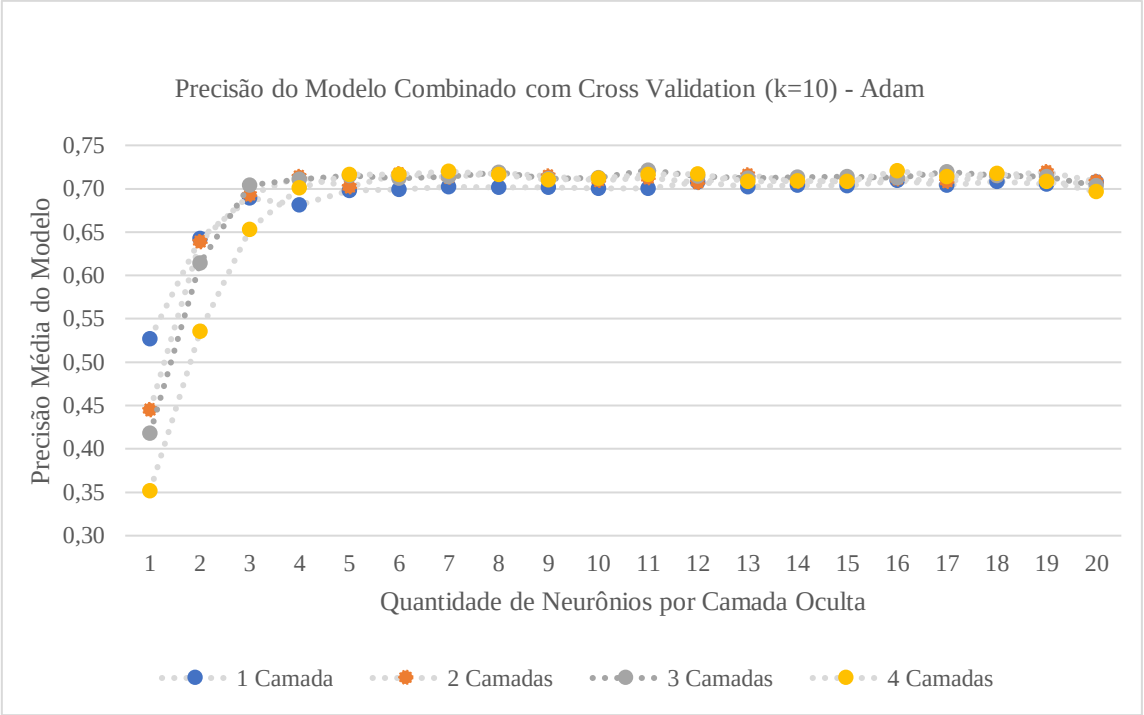
```
LEFT JOIN beecambio.tbl_profession p on p.id = c.profession_id
LEFT JOIN ticket_medio ON ticket_medio.id_customer = c.id
LEFT JOIN region ON region.id_customer = c.id
LEFT JOIN first_op_rem ON first_op_rem.id_customer = c.id
LEFT JOIN first_op_esp ON first_op_esp.id_customer = c.id
LEFT JOIN lim ON lim.id_customer = c.id
LEFT JOIN limites ON limites.id_customer = c.id

WHERE genre is not NULL
AND (ops_remessa is not NULL or ops_especie is not NULL)
AND c.created_at < '{end_date}'
AND c.created_at > '20170501';
    """.format(end_date='20180701')
```

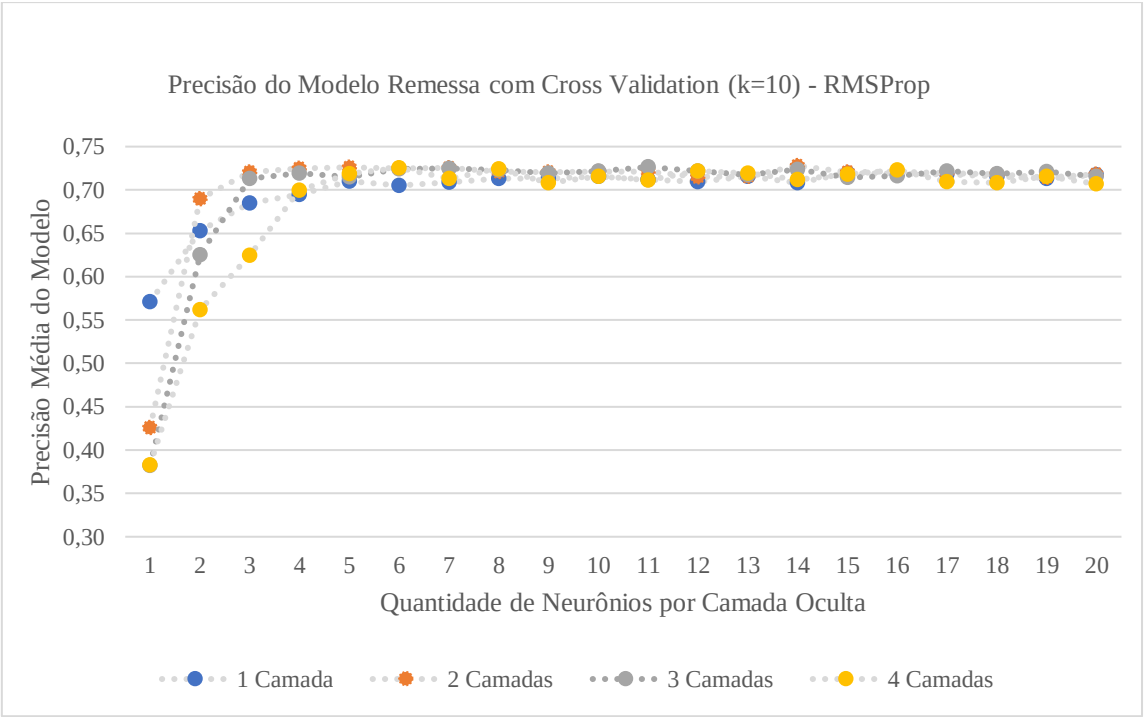
```
def email_visit_data(filter_site=None):
    return """
    SELECT user_id,
    MAX(CASE WHEN idsite = 5 THEN visitor_count_visits ELSE 0 END) as
visitor_count_visits_remessa,
    SUM(CASE WHEN idsite = 5 THEN visit_total_actions ELSE 0 END) as
visit_total_actions_remessa,
    SUM(CASE WHEN idsite = 5 THEN visit_total_interactions ELSE 0 END) as
visit_total_interactions_remessa,
    SUM(CASE WHEN idsite = 5 THEN visit_total_time ELSE 0 END) as
visit_total_time_remessa,
    MAX(CASE WHEN idsite = 6 THEN visitor_count_visits ELSE 0 END) as
visitor_count_visits_especie,
    SUM(CASE WHEN idsite = 6 THEN visit_total_actions ELSE 0 END) as
visit_total_actions_especie,
    SUM(CASE WHEN idsite = 6 THEN visit_total_interactions ELSE 0 END) as
visit_total_interactions_especie,
    SUM(CASE WHEN idsite = 6 THEN visit_total_time ELSE 0 END) as
visit_total_time_especie,
CASE WHEN SUM(CASE WHEN config_os = "WIN" AND idsite = 5 THEN 1 ELSE 0 END)
> 0 THEN 1 ELSE 0 END as windows_remessa,
CASE WHEN SUM(CASE WHEN config_os = "IOS" AND idsite = 5 THEN 1 ELSE 0 END)
> 0 THEN 1 ELSE 0 END as ios_remessa,
CASE WHEN SUM(CASE WHEN config_os = "AND" AND idsite = 5 THEN 1 ELSE 0 END)
> 0 THEN 1 ELSE 0 END as android_remessa,
CASE WHEN SUM(CASE WHEN config_os = "MAC" AND idsite = 5 THEN 1 ELSE 0 END)
> 0 THEN 1 ELSE 0 END as mac_remessa,
CASE WHEN SUM(CASE WHEN config_os = "WIN" AND idsite = 6 THEN 1 ELSE 0 END)
> 0 THEN 1 ELSE 0 END as windows_especie,
CASE WHEN SUM(CASE WHEN config_os = "IOS" AND idsite = 6 THEN 1 ELSE 0 END)
> 0 THEN 1 ELSE 0 END as ios_especie,
CASE WHEN SUM(CASE WHEN config_os = "AND" AND idsite = 6 THEN 1 ELSE 0 END)
> 0 THEN 1 ELSE 0 END as android_especie,
CASE WHEN SUM(CASE WHEN config_os = "MAC" AND idsite = 6 THEN 1 ELSE 0 END)
> 0 THEN 1 ELSE 0 END as mac_especie

FROM piwik_beetech.archive_120days_log_visit_20180711
WHERE user_id is not null AND user_id not like "%beetech%" AND user_id not
like "%beecambio%"
AND user_id not like "%beecapital%"
GROUP BY user_id ORDER BY SUM(visit_total_time) DESC;
    """
```

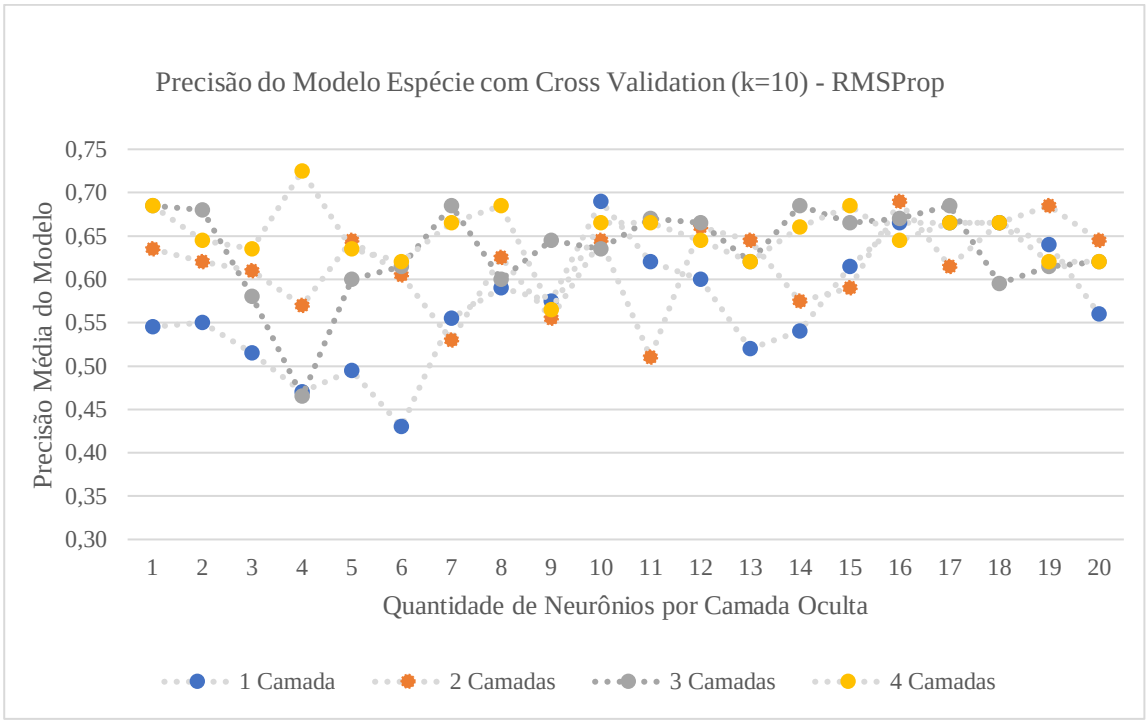
APÊNDICE D – RESULTADOS COMPLETOS DA PRECISÃO DOS MODELOS

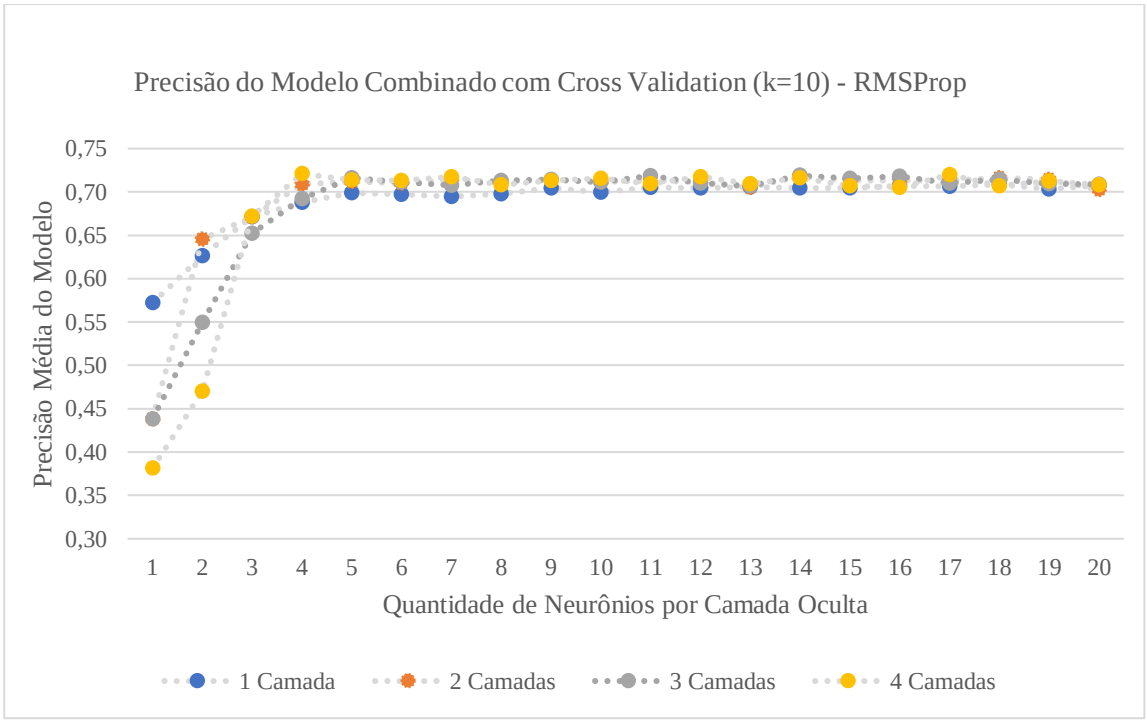





Precisão do Modelo Combinado com Cross Validation (k=10) - Adam
Precisão Média do Modelo
0,75
0,70
0,65
0,60
0,55
0,50
0,45
0,40
0,35
0,30
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Quantidade de Neurônios por Camada Oculta
1 Camada
2 Camadas
3 Camadas
4 Camadas


Precisão do Modelo Remessa com Cross Validation (k=10) - RMSProp
Precisão Média do Modelo
0,75
0,70
0,65
0,60
0,55
0,50
0,45
0,40
0,35
0,30
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Quantidade de Neurônios por Camada Oculta
1 Camada
2 Camadas
3 Camadas
4 Camadas


Precisão do Modelo Espécie com Cross Validation (k=10) - RMSProp
Precisão Média do Modelo
0,75
0,70
0,65
0,60
0,55
0,50
0,45
0,40
0,35
0,30
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Quantidade de Neurônios por Camada Oculta
1 Camada
2 Camadas
3 Camadas
4 Camadas


Precisão do Modelo Combinado com Cross Validation (k=10) - RMSProp
Precisão Média do Modelo
0,75
0,70
0,65
0,60
0,55
0,50
0,45
0,40
0,35
0,30
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Quantidade de Neurônios por Camada Oculta
1 Camada
2 Camadas
3 Camadas
4 Camadas