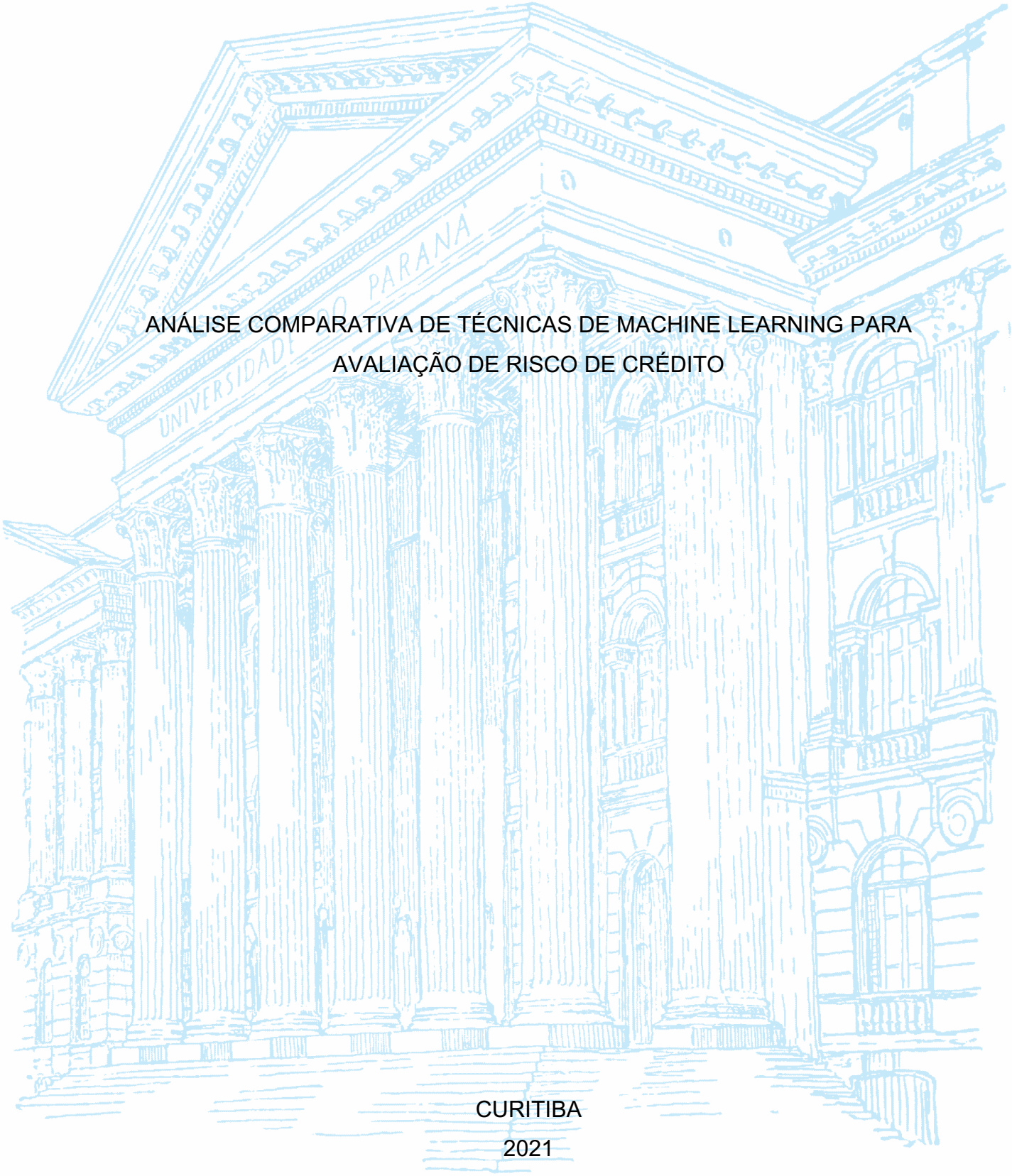
UNIVERSIDADE FEDERAL DO PARANÁ

MATEUS WILLIAM PEDROZO



ANÁLISE COMPARATIVA DE TÉCNICAS DE MACHINE LEARNING PARA AVALIAÇÃO DE RISCO DE CRÉDITO

CURITIBA

2021

MATEUS WILLIAM PEDROZO

ANÁLISE COMPARATIVA DE TÉCNICAS DE MACHINE LEARNING PARA AVALIAÇÃO DE RISCO DE CRÉDITO

Trabalho de Conclusão de Curso apresentado ao curso de Graduação em Engenharia de Produção, Setor de Tecnologia, Universidade Federal do Paraná, como requisito parcial à obtenção do título de Bacharel em Engenharia de Produção.

Orientadora: Profa. Dra. Mariana Kleina

CURITIBA

2021

RESUMO

A predição do comportamento dos clientes no mercado de crédito tem sido uma área importante na pesquisa de algoritmos de *machine learning* nos últimos anos. O crescimento exponencial nos dados gerados pelos clientes desafia os métodos tradicionais de previsão a absorverem cada vez mais informações para os modelos. O objetivo deste trabalho é comparar dois métodos de *machine learning*: redes neurais artificiais e *support vector machines*, visando identificar qual apresenta melhor desempenho na previsão de inadimplência de clientes. Ademais, esta pesquisa propõe comparar diferentes arquiteturas nos métodos testados para buscar a melhor organização. Por fim, os resultados são avaliados por meio de indicadores de performance, como acurácia, a eficiência dos modelos depende dos parâmetros escolhidos para cada método, variando entre 84,49% e 99,69%.

Palavras-chave: Inteligência Artificial. Risco de Crédito. Classificação.

ABSTRACT

The prediction of customers behavior in credit market has been an interesting area in the research of machine learning algorithms in recent years. In addition, the exponential growth in customer-generated data challenges traditional forecasting methods to absorb more information into models. Furthermore, this work proposes to compare two powerful machine learning algorithms, known as, Artficial Neural Networks and support vector machines Identify which has the best performance in predicting customer default. Furthermore, this work proposes to compare different architectures in the tested models in order to find the best organization. Finally, the results are evaluated using performance indicators such as accuracy, the efficiency of the models depends on the parameters chosen for each algorithm, ranging from 84,49% e 99,69%.

Keywords: Artificial Intelligence. Classification. Credit Market. Classification.

SUMÁRIO

1 INTRODUÇÃO

Estima-se que o volume de dados gerados no mundo está dobrando de tamanho a cada 2 anos e atingiu 4,4 trilhões de gigabytes em 2013 (IDC, 2013). Esse fenômeno tem impactado o mundo dos negócios, em especial, o segmento financeiro, que tem experimentado um expressivo crescimento da quantidade de informações. A posse dessa grande quantia de dados traz o desafio de gerenciar, armazenar e processar tudo o que se tem. Surge, então, a necessidade de técnicas e ferramentas para a análise desses dados, a fim de encontrar conhecimentos ocultos nas entrelinhas. Na maioria das vezes, pode-se encontrar padrões, difíceis de serem percebidos, que são muito úteis para as instituições, mas que não são encontrados sem o uso de ferramentas específicas para analisar milhares de linhas (FAYYAD, 1996).

Esse desafio tem impulsionado o aprimoramento de técnicas cada vez mais assertivas na comercialização dos produtos de crédito. Nesse segmento, os bancos captam recursos de agentes superavitários e entregam aos agentes deficitários, cobrando diferentes taxas de juros, assumindo para si os riscos e vislumbrando os lucros, conforme os agentes deficitários honrem com os pagamentos.

Nesse contexto, avaliar o risco de inadimplência, não somente mediante os critérios convencionais, mas também mediante a utilização de métodos estatísticos sofisticados, pode contribuir para a solidez de instituições financeiras e das empresas em geral. As abordagens de análise que utilizam aprendizado de máquina (do inglês *machine learning*) têm sido cada vez mais empregadas para avaliar o perfil dos clientes, principalmente, como uma resposta aos ambientes dinâmicos, nos quais decisões devem ser tomadas rapidamente.

A ideia por trás do aprendizado de máquina é o desenvolvimento de algoritmos que melhoram automaticamente com a experiência. Trata-se de uma área do conhecimento que se situa na interseção da ciência da computação e estatística, no centro da inteligência artificial e da ciência de dados, que está em rápida expansão na atualidade (JORDAN; MITCHELL, 2015).

O objetivo deste trabalho é comparar dois métodos de aprendizado de máquina, identificando qual apresenta melhor desempenho ao prever a inadimplência de uma base de clientes. Quanto aos métodos de aprendizado de máquina, avaliou-se os algoritmos redes neurais artificiais e *support vector machines*. Três medidas de

erro foram adotadas na comparação do desempenho dos modelos: média do quadrado dos erros, média absoluta dos erros e média absoluta percentual dos erros. Foi analisada uma base de dados composta de pessoas físicas com dois mil clientes que contém características financeiras dos clientes e o histórico de inadimplência.

1.1 OBJETIVO

Investigar a acurácia, em termos da capacidade preditiva, de redes neurais artificiais e máquina de vetores de suporte para a previsão de ocorrência de perdas financeiras, devido ao não cumprimento das obrigações por parte do tomador de crédito.

1.2 OBJETIVOS ESPECÍFICOS

O desenvolvimento do estudo envolve:

- Tratar dados inconsistentes da base que possuem o histórico dos clientes;
- Aplicar redes neurais artificiais e testar seus parâmetros utilizando bibliotecas *python*;
- Aplicar *support vector machines* e testar seus parâmetros utilizando bibliotecas *python*;
- Comparar, mediante análise da acurácia, os resultados dos métodos quantitativos aplicados.

2 REFERENCIAL TEÓRICO

O risco de crédito é definido como o risco de perda resultante de falha entre alguma das contrapartes no cumprimento das obrigações (QU, 2008). O risco inerente a esse tipo de operação é denominado de inadimplência e ocorre de diferentes formas: pela pré-disposição do agente não identificada pelo banco, por eventos adversos fora do controle do agente, por eventos conjunturais do país ou por erro do banco na análise do risco.

As consequências da inadimplência para o sistema de financeiro vão desde um aumento pontual da taxa de juros cobrada nas operações de uma instituição específica até a desestabilização do sistema financeiro de diversos países, também conhecido como crise sistêmica (ROCHET, 2007). A partir desse ponto, ocorrem danos à chamada economia real, aumentando a probabilidade de quebra de bancos, corrida de depositantes para retirada de recursos do sistema financeiro e retração dos agentes superavitários em fuga ao risco.

Diante disso, avaliar se o cliente irá deixar de cumprir com suas obrigações financeiras é de extrema importância e pode causar um grande impacto no balanço da Instituição. Cada vez mais soluções vêm sendo desenvolvidas e aprimoradas visando minimizar o risco de *default.*

Default, por sua vez, é o termo utilizado para indicar o não cumprimento das obrigações e/ou condições de um empréstimo (como financiamentos ou dívidas de cartão de crédito). Normalmente, o principal motivo para o descumprimento das condições de pagamento é a incapacidade financeira do cliente.

A modelagem de sistemas internos de classificação e controle de risco envolve o cálculo de probabilidades de inadimplência. É comum, o uso de abordagens estatísticas utilizando modelos paramétricos e não paramétricos, dos quais pode-se citar: análise de regressão, análise discriminante, regressão Logit e Probit. Diversos pesquisadores têm investigado o uso de técnicas de inteligência computacional: Redes Neurais Artificiais, Árvores de Decisão, Lógica Difusa, Máquinas de Vetores de Suporte, para a identificação de padrões de características dos tomadores de crédito e classificação de risco nas operações realizadas.

2.1 REDES NEURAIS

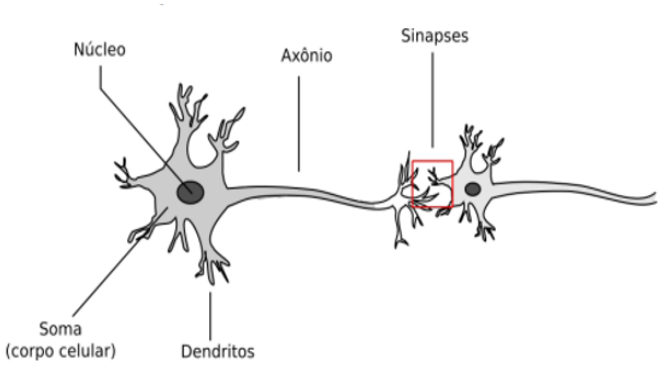
O sistema nervoso é responsável pela maioria das funções de controle em um organismo, coordenando e regulando as atividades corporais. Estas funções só são possíveis pelo aprendizado contínuo, possibilitando ao corpo se adaptar ao meio ambiente com sucesso. A célula mais representativa e importante do sistema nervoso é o neurônio. No cérebro humano, por exemplo, há cerca de 10 bilhões de neurônios e aproximadamente 60 trilhões de conexões neuronais entre si (HAYKIN, 2001).

2.1.1 Neurônio

O neurônio é a unidade funcional do sistema nervoso composto de muitas entradas e saídas. Os neurônios comunicam-se por meio de sinapses que conectam os dendritos aos axônios de outras células nervosas. Os sinais que chegam desses axônios são impulsos elétricos chamados de impulsos nervosos ou potenciais pulsos de ação e constituem-se em informações geradas pelo processamento neural na forma de uma saída de impulso neural em seus axônios (GONZÁLEZ, 2002).

Anatomicamente, o neurônio é formado por dendrito, corpo celular e axônio, conforme pode ser observado na Figura 1. Os dendritos têm por função receber os estímulos transmitidos pelos outros neurônios, já o corpo celular é responsável por coletar e combinar informações vindas de outras células neurais O axônio é constituído de uma fibra tubular que pode alcançar até alguns metros e é responsável por transmitir os estímulos para outras células. A transmissão ocorre apenas no sentido do dendrito ao axônio.

Figura 1 – Neurônio Biológico



Fonte: Azevedo (2016)

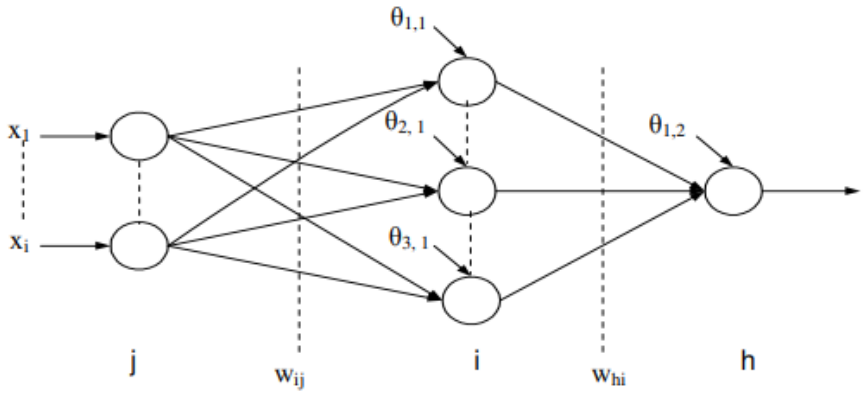
2.1.2 Redes Neurais Artificiais

Segundo Beale (2000 *apud* MULLER, 1996), as redes neurais artificiais (RNA) têm como objetivo replicar os princípios básicos de manipulação de informação do cérebro humano e aplicar esse conhecimento na resolução de problemas que exigem aprendizado a partir da experiência.

Os modelos de RNA se diferenciam dos modelos tradicionais de previsão, envolvendo algoritmos de aprendizado. Tais algoritmos buscam imitar a estrutura de interconexões do sistema nervoso, com intuito de compreender e replicar o padrão de comportamento dos dados analisados, da maneira mais eficiente possível (TURBAN, 1993).

As RNAs podem ser interpretadas como modelos de regressão não linear, que permite ao analista ajustar grande número de parâmetros e testar diferentes configurações para um ajuste (CHATFIELD, 1996). Essas diferentes variáveis podem ser interpretadas como formas funcionais alternativas para um ajuste adequado da rede em termos do número de suas camadas e do número de neurônios em cada camada (FERNANDES; PORTUGAL, 1995). O elemento básico de uma RNA é o neurônio artificial (Figura 2). Sua estrutura busca reproduzir a estrutura do neurônio biológico, como proposto por McCulloch e Pitts (1943).

Figura 2 – Multilayer Perceptron



Fonte: adaptado de Neto (2014)

O RNA é constituído de diferentes camadas (PORTUGAL e KOHZADI, 1995). Essencialmente, existem três tipos de camadas, assim como ilustrado na Figura 2: camada de entrada representada pela por j, intermediária (também chamada de oculta ou escondida) representada por i e camada de saída representada por h.

A primeira camada da rede, denominada de camada entrada, recebe os insumos externos à rede sem nenhum procedimento computacional. As camadas ocultas recebem os dados ponderados pela camada de entrada e são constituídas pelos neurônios que são responsáveis pela extração das características associadas ao sistema a ser inferido. A camada de saída h recebe informações da camada oculta e fornece o produto da rede. A rede descrita na Figura 2 é do tipo *feedforward* (transmissão em um só sentido entre os nós), qual é denominado de *perceptron* de múltiplas camadas ou seu termo em inglês *multilayer perceptron* (MLP). É um exemplo simplificado de RNA, posto que as informações podem movimentar-se em várias direções em outras arquiteturas.

Barone (2003) afirma que um neurônio artificial pode ser segmentado pelos seguintes elementos: um conjunto de valores de entrada, um conjunto de pesos, um conjunto de funções de ativação, um fator corretivo chamado Bias e uma saída.

Cada neurônio é capaz de processar os valores de entrada e transformá-los em um sinal de saída. O estado de ativação do neurônio em questão é calculado a partir da aplicação de uma função de limiar ao valor de entrada fornecido ao neurônio, ou seja, a somatória dos valores de ativação dos neurônios precedentes, multiplicados pelos respectivos pesos. Conforme a RNA ilustrada na Figura 2, uma unidade i recebe os sinais de entrada e os agrega baseada em uma função de entrada. Matematicamente, esse procedimento equivale a equação (1).

$$i_{pi} = \sum_{j=0}^{p} w_{ij} x_{pj} + \theta_i \qquad (1)$$

com $p = 1,\ldots,(m+k)$, $i = 1,\ldots,k*$ e $j = 1,\ldots,n$, na qual i_{pi} é a entrada da unidade i para o padrão p; i é o número de unidades na camada escondida; w_{ij} é a conexão peso entre as unidades i e j; x_{pj} são as entradas do padrão p e θ_{ni} são as bias das unidades n na camada i.

Essa função de entrada gera um sinal de saída, a_{pj}, para o padrão p, utilizando a função de ativação sigmoidal que tem como objetivo calcular o estado final, ou de saída, do neurônio, determinando o quanto o nível de ativação do neurônio está acima de um valor limiar para que, assim, ela possa determinar se o neurônio é ou não ativado. Matematicamente, esse procedimento pode ser representado como na equação (2).

$$a_{pj} = \frac{1}{(1 + e^{-i_{pi}})} \tag{2}$$

Esses sinais de saída são, então, enviados à unidade da camada h, ilustrada da Figura 2, que os agrega conforme equação (3).

$$i_{ph} = \sum_{j=0}^{p} w_{hi} a_{pi} + \theta_{hn} \tag{3}$$

Gerando a saída demonstrada na equação (4).

$$a_{ph} = \frac{1}{(1 + e^{-i_{ph}})} \tag{4}$$

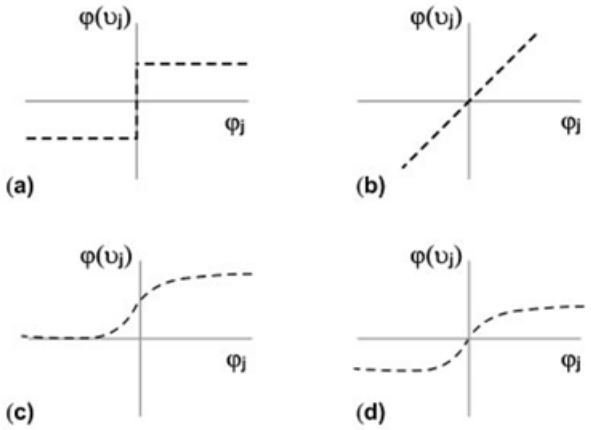
2.1.3 Funções de Ativação

As funções de ativação, também chamadas de funções restritivas, são um elemento fundamental das redes neurais artificiais, uma vez que elas basicamente decidem se um neurônio deve ser ativado ou não. Ou seja, se a informação que o neurônio está recebendo é relevante para o processo ou deve ser ignorada.

As funções de ativação podem ser lineares ou não lineares (NETO, 2014). A função linear tem como objetivo associar as características lineares dos valores do insumo de entrada, seus resultados podem ser aproximados aos resultados de uma regressão linear. A função não linear busca captar as características não lineares das variáveis de entrada (NETO, 2014). A Figura 3 destaca 4 importantes tipos de funções

de ativação: a função linear, a função logística, a função degrau e a função tangente hiperbólica.

Figura 3 – Representação Gráfica De Diferentes Funções De Ativação: (A) Função Degrau; (B) Função Linear; (C) Função Logística; (D) Função Tangente Hiperbólica



Fonte: Braga (2000)

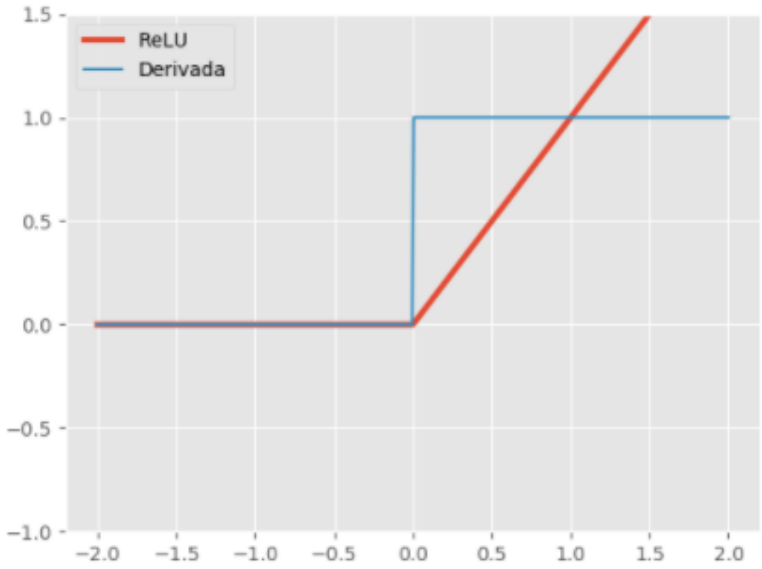
2.1.4 Funções de Ativação ReLU

Redes com a função ReLU são fáceis de otimizar, já que a ReLU é extremamente parecida com a função identidade. A única diferença é que a ReLU produz zero em metade do seu domínio, conforme a Figura 4. Como consequência, as derivadas se mantêm grandes enquanto a unidade estiver ativa (KOWALCZYK, 2017).

A função ReLU retorna zero para todos os valores negativos, e o próprio valor para valores positivos. É uma função computacionalmente leve, entretanto, não é centrada em zero. Como seu resultado é zero para valores negativos, ela tende a “apagar” alguns neurônios durante um passo forward, o que aumenta a velocidade do treinamento, mas por outro pode fazer com que esses neurônios “morram” e não aprendam nada se eles só receberem valores negativos. Além disso, ela pode produzir ativações extrapoladas já que não possui um limite positivo (KOWALCZYK, 2017).

A ativação ReLU é muito mais eficiente do que as funções sigmoidais vistas anteriormente e é uma das descobertas que contribuiu de forma significante para a recente popularidade de *Deep Learning*. Essa não linearidade é um ótimo exemplo de como a simplicidade pode ser extremamente poderosa (KOWALCZYK, 2017).

Figura 4 – Representação Gráfica Da Função Relu



Fonte: Kowalczyk (2017)

2.1.5 Aprendizado de uma Rede Neural Artificial

De acordo com Freeman (1992 *apud* MUELLER, 1996), a propriedade mais importante de uma RNA é o seu potencial de aprendizado a partir do ambiente no qual ela está inserida. Aprendizagem é o processo pelo qual os parâmetros de uma rede neural são adaptados por meio de um estímulo contínuo do ambiente, no qual a rede está operando, sendo o tipo específico de aprendizagem realizada definido pela maneira particular como ocorrem os ajustes realizados nos parâmetros (HAYKIN, 1994). O objetivo do treinamento consiste em atribuir os pesos sinápticos com valores adequados, de modo a produzir o conjunto de saídas desejadas ou ao menos consistentes com um intervalo de erro estabelecido.

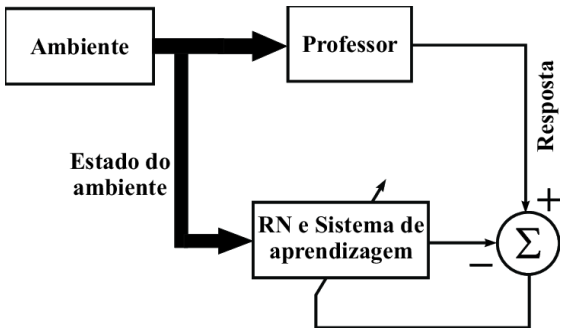
De acordo com Trippi e Turban (1993): O processo de aprendizagem pode ser simplificado da seguinte forma:

1 - Compute a saída;
2 - Compare a saída com as respostas desejadas;
3 - Ajuste os pesos e repita o processo.

Os dados utilizados para treinamento da rede devem ser significativos e cobrir amplamente o domínio do problema para que o ambiente real seja bem simulado, e não apenas envolver os casos de operações normais ou rotineiras, mas também incorporar casos de exceção e de situações limites.

O aprendizado de uma rede neural pode ser segmentado em dois tipos: aprendizado supervisionado e aprendizado não supervisionado (BRAGA, 2000). Nesse tipo de aprendizagem, ilustrado na Figura 5, é utilizado um agente externo que indica à rede a resposta desejada para o padrão de entrada. O objetivo é ajustar os parâmetros da rede, de forma a encontrar uma ligação entre os pares de entrada e saída fornecidos. O “professor” indica explicitamente um comportamento bom ou ruim para a rede, visando direcionar o processo de treinamento.

Figura 5 – Aprendizado Supervisionado



Fonte: adaptado de Haykin (2001)

No aprendizado não supervisionado, não há resultados pré-estabelecidos para acompanhar o processo de aprendizado. Para esses algoritmos, somente os padrões de entrada estão disponíveis para a rede. A partir do momento em que a rede estabelece uma harmonia com as regularidades estatísticas da entrada de dados,

desenvolve-se nela uma habilidade de formar representações internas para codificar características da entrada e saída.

2.1.6 Aprendizado por *Backpropagation*

O modelo de aprendizado supervisionado mais popular na literatura é modelo de retropropagação *backpropagation*. Esse método de aprendizado conduz o erro no valor de saída para trás ao longo do sistema, ajustando, dessa forma, os pesos de cada neurônio desde a última camada da RNA até a primeira camada (HAIR, 2005).

Rumelhard, Hinton e Williams foram os criadores do algoritmo *backpropagation* em 1986 a partir da generalização da regra de aprendizado Widrow-Hoff, que fora introduzida em 1960-1962 para redes do tipo *feedforward perceptron* (ZURADA, 1992; HAYKIN, 1994).

É o vetor de pesos w demonstrado na equação (1) que constitui o que a rede neural "sabe" e determina como ela responderá a qualquer entrada arbitrária do meio ambiente. O algoritmo de aprendizagem *backpropagation* procurará por meio do espaço de W por um conjunto de pesos oferecendo o melhor ajuste para os pesos apresentados no início do processo. O algoritmo consiste de duas fases (propagações).

Na propagação *forward*, os $p = (m + k)$ padrões descritos por suas coordenadas x_{pj} alimentam a rede conforme as equações (3) a (4). O valor de saída obtido para o padrão p, a_{ph} é comparado com o valor de saída desejado para o padrão p, d_p, calculando-se o erro quadrático demonstrada na equação (5).

$$E = \frac{1}{2}\sum_{p=1}^{m+k}(d_p - a_{ph})^2 \tag{5}$$

Na segunda fase, a propagação *backward* executa um gradiente descendente em W para encontrar a solução ótima. A direção e magnitude $\Delta W_{i,j}$ podem ser calculadas pela equação (6).

i. Variação em W_{hi}'s

$$\Delta_p W_{hi} = \gamma * a_{ph}(1 - a_{ph}).a_{pi} * (d_p - a_{ph}) \tag{6}$$

Considerando na variação dos pesos para o padrão atual t a troca de pesos obtida para o padrão anterior (t – 1), a fim de alcançar o mínimo mais rapidamente demonstrado na equação (7).

$$\Delta_p W_{hi}(t) = \gamma * a_{ph}(1 - a_{ph}).a_{pi} * (d_p - a_{ph}) + \alpha * \Delta_p W_{hi}(t-1) \tag{7}$$

onde α é a constante que determina o efeito na troca de pesos em (t-1), também chamada de taxa de aprendizado. Então, os pesos entre as camadas escondidas e a camada de saída, em t, podem ser determinados utilizando-se a equação (8).

$$\Delta_p W_{ij}(t) = \gamma * (d_p - a_{ph}).a_{ph} * (1 - a_{ph}) * W_{hi} * a_{pi}(1 - a_{pi}) * x_{pj} \tag{8}$$

Considerando na variação dos pesos para o padrão atual t a troca de pesos obtida para o padrão anterior (t – 1), tem-se:

$$\Delta_p W_{ij}(t) = \gamma * (d_p - a_{ph}).a_{ph} * (1 - a_{ph}) * W_{hi} * a_{pi}(1 - a_{pi}) * x_{pj} * \alpha * \Delta_p W_{ij}(t-1) \tag{9}$$

Então, os pesos entre a camada de entrada e a camada escondida, em t, podem ser determinados pela equação (10).

$$W_{ij}(t) = W_{ij}(t-1) + \Delta_p W_{ij}(t) \tag{10}$$

2.2 TREINAMENTO DE REDES NEURAIS

O processo de treinamento de uma RNA é parte fundamental de qualquer desenvolvimento nesse ramo. Essa etapa exige do desenvolvedor uma seleção adequada dos parâmetros livres da rede para o conjunto de dados de treinamento.

Martinelli (1999) afirma que para obtenção de um bom resultado de uma Rede Neural, “[...] devem ser feitas várias divisões do mesmo conjunto de dados em conjunto de treinamento e de testes. Este método, chamado de *cross validation*, fornece uma estimativa mais realista do erro a ser cometido pelo método de classificação utilizado”.

Essa técnica estatística também chamada de “Validação Cruzada” pode ser resumida, segundo Kohavi (1995), como uma forma simples de validar um modelo que envolve separar uma parte dos dados de treinamento e usá-los para obter previsões do modelo treinado sobre o restante dos dados. A estimativa de erro, em seguida, informa como o modelo está se saindo em dados não vistos ou no conjunto de validação. Este é um tipo simples de técnica de validação cruzada, também conhecida como método *holdout*. O problema com esse método é que não é certo que o set de validação separado (*holdhout*) seja representativo do total da base de dados.

2.2.1 Método k-fold

No método *K-fold* (BRURMAN, 1989), os dados são seccionados em *k* subconjuntos. Em seguida, o método *holdout* é repetido *k* vezes, de tal forma que, a cada vez, um dos *k* subconjuntos é usado como set de validação e os outros subconjuntos *k-1* são colocados juntos para formar um set de treinamento. A estimativa de erro é calculada com base em todas as *k* tentativas para obter a eficácia total do modelo.

Segundo Borra e Ciaccio (2010), cada observação integra o set de validação exatamente uma vez e integra o set de treinamento k-1 vezes. Isso reduz significativamente o bias (viés), já que utiliza-se a maioria dos dados para treinamento e reduz significativamente a variância, pois a maioria dos dados também está sendo usada no set de validação. Como regra geral e evidência empírica, K = 5 ou 10 são boas opções, mas nada é fixo e pode-se usar qualquer valor.

2.2.2 Máquina de Vetores de Suporte (SVM)

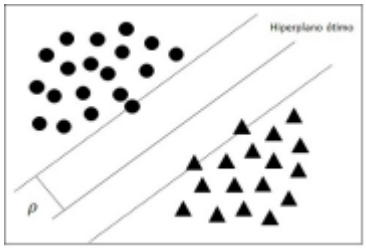
A Máquina de Vetores de Suporte (SVM do termo em inglês *Support Vector Machine*) proposta por Vapnik (1995) é um algoritmo de aprendizado supervisionado utilizado na classificação e regressão de dados para conjuntos linearmente ou não linearmente separáveis, por meio da escolha de um hiperplano ótimo de forma que a separação dos conjuntos de dados seja máxima. Um hiperplano é um objeto matemático que pode ser um ponto, caso esteja-se interpretando dados em uma dimensão ($\mathbb{R}$), uma reta, para duas dimensões ($\mathbb{R}^2$), um plano para três dimensões ($\mathbb{R}^3$) e assim sucessivamente (KOWALCZYK, 2017).

2.2.3 Hiperplanos de separação

Uma SVM constrói um classificador de acordo com um conjunto de padrões por ele identificados nos exemplos de treinamento, no qual a classificação é conhecida.

Considerando o exemplo da Figura 6, nela existe um conjunto de classificadores lineares que separam duas classes, mas apenas um (em destaque) que maximiza a margem de separação (distância da instância mais próxima ao hiperplano de separação das duas classes em questão). O hiperplano com margem máxima é chamado de hiperplano ótimo, que será o objeto de busca do treinamento do classificador (GUNN, 1998). Um hiperplano com margem máxima é considerado ótimo se consegue separar um conjunto de dados e ainda maximizar a distância entre os vetores, como pode ser observado a seguir.

Figura 6 – Hiperplano Ótimo Separando os Dados com Máxima Margem



Fonte: adaptado de ABE (2005)

2.2.4 SVM com margens Rígidas

A formulação matemática, baseada em ABE (2005) e implementada nos algoritmos de SVM, é dada por um conjunto de treinamento x_i com $(i = 1, 2 \ldots, D)$ em um problema que consiste em duas classes linearmente separáveis w_1 e w_2, no qual cada amostra associa-se $y_i = 1$, quando $x_i \in w_1$ e $y_i = -1$, quando $x_i \in w_2$. Em termos gerais,

$$D(x) = \sum_{i=1}^{D} w_i x_i + b \tag{11}$$

equivalente, em termos de produto interno:

$$D(x) = \boldsymbol{w^T x} + b \tag{12}$$

onde w é um vetor d-dimensional (pesos) e b um termo independente. Suponha que os exemplos de treinamento são linearmente separáveis, isto é, satisfazem as seguintes restrições:

$$x_i . w + b \geq 1 \, para \, y_i = 1 \tag{13}$$

$$x_i . w + b \leq -1 \, para \, y_i = -1 \tag{14}$$

Combinando as igualdades, pode-se obter:

$$y_i(x_i . w + b) - 1 \geq 0, para \; i = 1,2, \ldots, D \tag{15}$$

Considerando exemplos, nos quais se dá a igualdade na equação (15), esses são pontos sobre o hiperplano $x_i . w + b = 1$ com normal w e distância perpendicular a origem de $\frac{|1-b|}{\|w\|}$ e para o caso de $x_i . w + b = -1$ com normal w e distância perpendicular a origem de $\frac{|-1-b|}{\|w\|}$.

Assim, a largura da margem possui valor $\frac{2}{\|w\|}$ e para maximizar os dois hiperplanos que geram essa margem, minimiza-se $\|w\|^2$, gerando um problema de otimização quadrática sujeito às restrições definidas na equação (15).

$$min_{x,b} = \frac{1}{2}\|w\|^2 \tag{16}$$

A inclusão das restrições no problema de minimização pode ser resolvida por meio da técnica de multiplicadores de Lagrange, pois sem essa técnica, a minimização seria muito trabalhosa, dado que w forma um produto escalar, como visto na equação (9).

A formulação lagrangeana é dada pela introdução de $a_i, (i = 1, 2, \dots, D)$, um para cada uma das restrições definidas na equação (15) e subtraindo o resultado da função objetivo definida, obtendo:

$$L(a, w, \beta) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{D} a_i[y_i(x_i.w + \beta) - 1] \tag{17}$$

É necessário minimizar a Equação (17) com relação a w e β e com o resultado restante maximizar com relação a α ≥ 0, conhecido como um problema de otimização dual.

Em um ponto ótimo, as seguintes equações de ponto de sela são:

$$\frac{\partial L}{\partial b} = 0 \text{ e } \frac{\partial L}{\partial w} = 0 \tag{18}$$

Gerando respectivamente,

$$\sum_{i=1}^{D} a_i y_i = 0 \ e \ w = \sum_{i=1}^{D} a_i y_i x_i \tag{19}$$

Substituindo esses resultados na formulação lagrangeana, maximiza-se:

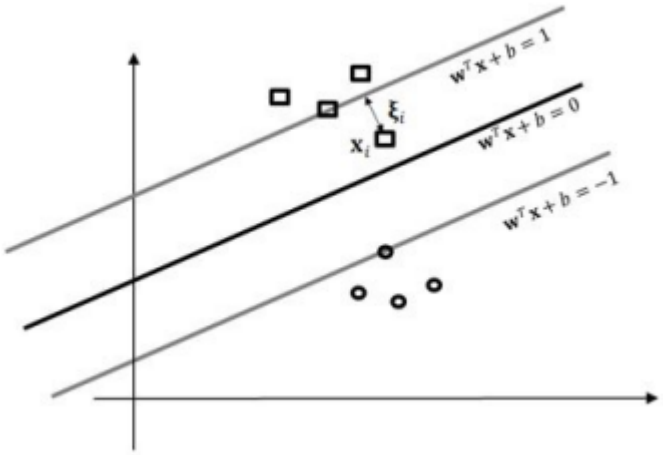
$$max_a\, W(a) = \sum_{i=1}^{D} a_i - \frac{1}{2}\sum_{j=1}^{D} a_i\, a_j y_i y_j \left(x_i * x_j\right)$$
$$sujeito\ a \qquad a_i \geq 0, \qquad i = 1,2, \ldots D \tag{20}$$
$$\sum_{i=1}^{D} a_i y_i = 0$$

Para a resolução do problema de maximização, utiliza-se o método gradiente descente (LORENA; CARVALHO, 2007). Pode-se mostrar que o SVM apresenta vantagens com respeito a classificadores convencionais, especialmente quando o número de amostras de treinamento é pequeno e a dimensionalidade dos dados é grande, devido ao fato de que os classificadores convencionais não têm mecanismos para maximizar a margem (distância entre os dois hiperplanos extremos). A maximização da margem permite aumentar a capacidade de generalização do classificador (ABE, 2005).

2.2.5 SVM com Margem Suave

O problema de otimização da equação (16) possui solução somente no caso das amostras x_i pertencerem a duas classes linearmente separáveis. Em situações reais, é pouco provável que duas classes sejam separáveis por um hiperplano no seu espaço original. O caso das classes não linearmente separáveis é tratado de forma idêntica, sendo, porém, necessário introduzir uma penalização às observações que se encontram do lado errado do hiperplano. Os hiperplanos que introduzem essa penalização são conhecidos como hiperplanos de margem suave, conforme demonstrado na Figura 7. No caso não separável, permite-se que existam amostras de treinamento dentro da margem máxima. A nova definição dos hiperplanos de suporte é obtida introduzindo a variável de folga ξ_i para cada ponto:

Figura 7 – Um Hiperplano Separador com Custos ξ_i Associados a Amostras de Treinamento Incorretamente Classificadas



Fonte: adaptado de Hamel (2009).

$$x_i . w^T + b \geq 1 - \xi_i \, para \, x_i \in w_1 \, (y_i = 1) \tag{21}$$

$$x_i . w^T + b \geq -1 + \xi_i \, para \, x_i \in w_2 \, (y_i = -1) \tag{22}$$

$$\xi_i \geq 0 \, \forall \, i \tag{23}$$

O procedimento de suavização da margem do classificador linear permite que alguns dados de treinamento permaneçam entre os hiperplanos de suporte. Além disso, permite também a ocorrência de alguns erros de classificação.

Para o caso onde $0 < \xi_i < 1$, a amostra correspondente não terá margem máxima, mas será rotulada corretamente. No caso de $\xi_i \geq 1$, a amostra x_i será rotulada erroneamente. Para que ocorra um erro é necessário que o respectivo ξ_i seja maior que 1, assim $\sum_i \xi_i$ é o limite superior para o número de erros de treinamento. Agora, o problema de otimização deve minimizar os erros de treinamento. Assim, modifica-se a função objetivo para minimizar:

$$Q(w, b, \alpha) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{D} \xi_i \tag{24}$$

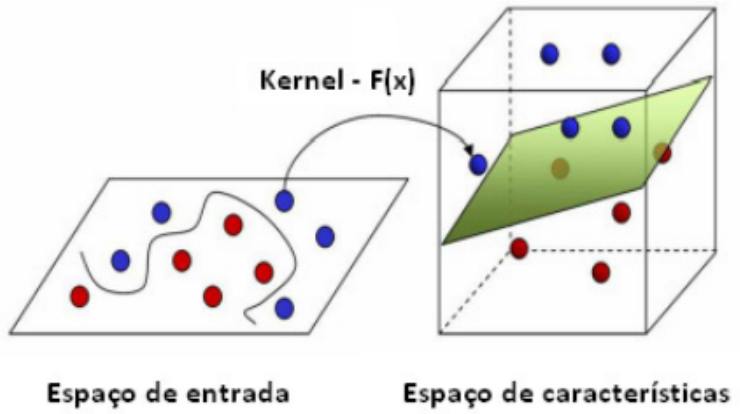
Onde C é um parâmetro a ser escolhido pelo usuário, quanto maior o valor do parâmetro C, maior será a penalização associada aos erros cometidos. A constante C é conhecida como “parâmetro de margem” e estabelece a importância relativa das duas parcelas do lado direito da igualdade da equação (24) nesse processo de minimização. A minimização de $\|w\|^2$ resulta na maximização da margem, enquanto que a minimização da segunda parcela $\sum_{i=1}^{D} \xi_i$ resulta na minimização do erro de classificação.

Ainda segundo Abe (2005), a única diferença entre o SVM de margem suave e o SVM de margem rígida é que a_i não pode exceder C. A função de decisão é a mesma para o caso de margem rígida.

2.2.9 Funções de Kernel

As funções de kernel têm a finalidade de projetar os vetores de características de entrada em um espaço de características de alta dimensão para classificação de problemas que se encontram em espaços não linearmente separáveis. Isso é feito, pois à medida que se aumenta o espaço da dimensão do problema, aumenta também a probabilidade desse problema se tornar linearmente separável em relação a um espaço de baixa dimensão. Entretanto, para obter-se uma boa distribuição para esse tipo de problema, é necessário um conjunto de treinamento com um elevado número de instâncias (GONÇALVES, 2010). A Figura 8 mostra o processo de transformação de um domínio não linearmente separável, em um problema linearmente separável por meio do aumento da dimensão, no qual é feito um mapeamento por uma função de kernel $F(x)$.

Figura 8 – Transformação de um Problema Não Linearmente Separável em um Problema Linearmente Separável



Fonte: Rebelo (2008)

Uma função é definida como sendo uma função de kernel se obedecer à seguinte teoria de Hilbert (*RKHS – Reproducing Kernel Hilbert Space*):

$$K(x_i, x_j) = < \phi(x_i) * \phi(x_j) > \tag{25}$$

Onde $\varphi(.)$ deve pertencer a um domínio no qual seja possível o cálculo do produto interno. Essas funções também satisfazem as condições do Teorema de Mercer (GONÇALVES, 2010). Teorema de Mercer, por sua vez, ocorre quando uma determinada função é definida como sendo de kernel se a matriz K é positivamente definida (autovalores maiores que zero), onde K é obtida por:

$$K = K_{ij} = K(x_i, x_j) \tag{26}$$

Essas funções de kernel também são conhecidas como Kernels de Mercer. Ressalta-se, aqui, que existem vários tipos de kernels, os quais podem ser usados, porém as mais conhecidas são mostradas na Figura 9.

Figura 9 – Principais Kernels

Tipo de Kernel	Função $\mathcal{K}(\mathbf{x_i}, \mathbf{x_j})$	Tipo do Classificador
Polinomial	$(\langle \mathbf{x_i} \cdot \mathbf{x_j} \rangle + 1)^p$	Máquina de aprendizagem polinomial
Gaussiano (ou RBF)	$\exp\left(-\frac{\|\mathbf{x_i} - \mathbf{x_j}\|^2}{2\sigma^2}\right)$	Rede RBF
Sigmoidal	$\tanh(\beta_0 \langle \mathbf{x_i} \cdot \mathbf{x_j} \rangle) + \beta_1$	Perceptron de duas camadas

Fonte: Rebelo (2008)

Segundo Abe (2005), a função de kernel RBF, demonstrada na Figura 9, possui dois parâmetros fundamentais para o processo de classificação das características, o C e o γ. O primeiro e responsável por determinar o melhor ponto para separação entre duas classes, enquanto que o segundo determina a largura do kernel ou a influência dos vetores de suporte. A combinação desses parâmetros influencia diretamente o resultado da classificação, porém encontrar os valores ideais pode ser complexo ou demandar muito tempo no ajuste ideal.

2.2.10 Indicadores de Performance dos modelos

A implementação de modelos quantitativos de previsão exige esforço para mensurar o desempenho das modelagens, com o intuito de identificar quão acurada é essa previsão. A literatura corrente demonstra uma variedade de métodos de previsão, porém, para chegar ao que apresenta o melhor resultado em comparação aos demais, é necessário um estudo sobre os erros que os modelos apresentam em relação às observações reais (PELLEGRINI; FOGLIATTO, 2001).

Para avaliar o desempenho dos métodos analisados neste estudo, foi utilizado a acurácia (ACC) dos modelos. Apesar de não serem comuns em estudos que se utilizam de métodos estatísticos convencionais, tais métricas são amplamente utilizadas quando se trata da avaliação de modelos de *machine learning.*

A ACC é uma medida da precisão do modelo em comparação aos dados gerais. Ela é calculada como a razão entre as unidades corretamente classificadas, verdadeiro positivo (TP) e falso positivo (FP) e o número total de previsões feitas

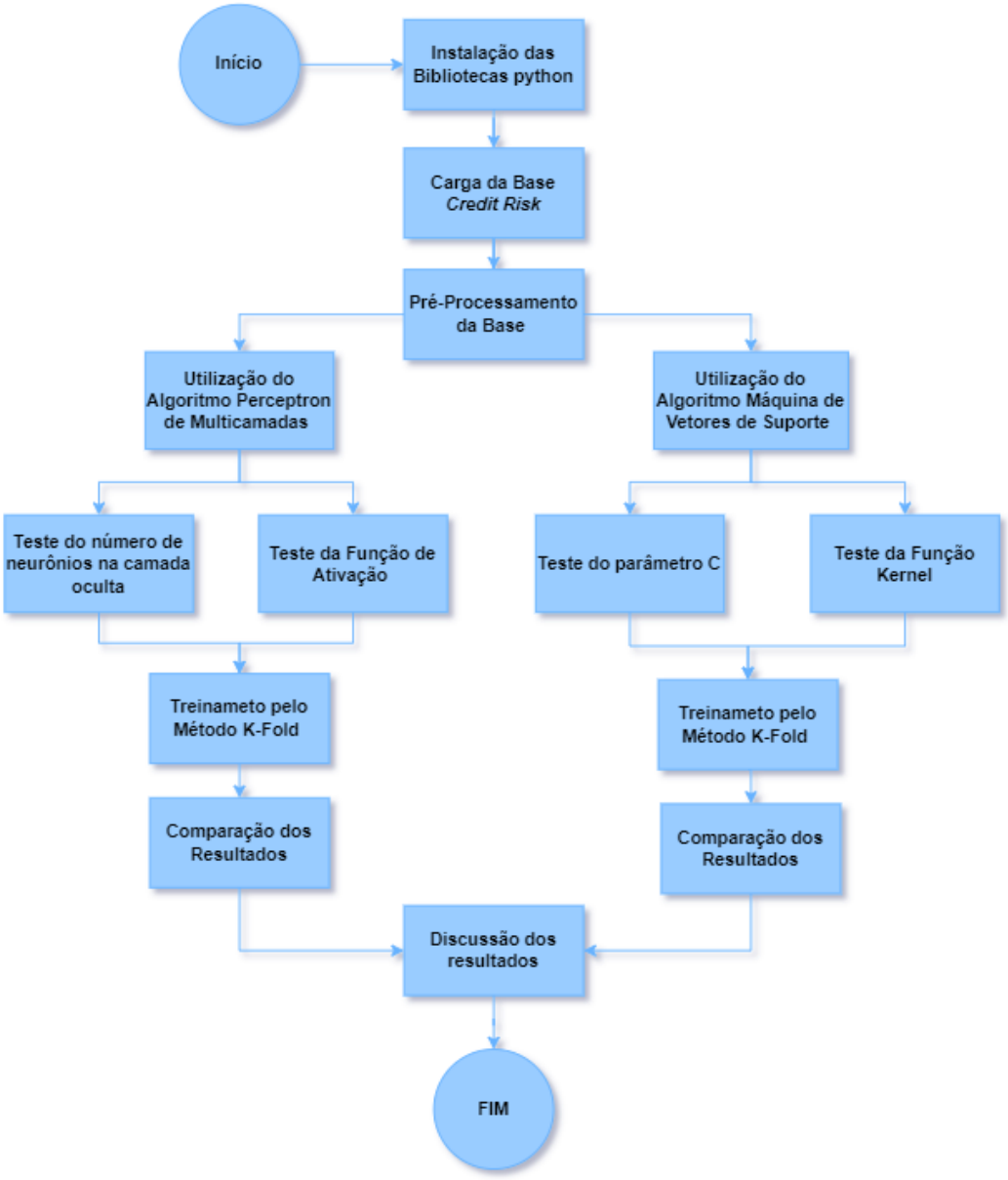
pelos classificadores, considerando os falsos negativos (FN) e os verdadeiros negativos (TN). ACC pode ser calculada conforme a equação (27) e quanto maior for a ACC, melhor a capacidade de predição do algoritmo.

$$ACC = \frac{TP + TN}{TP + FP + FN + TN} \tag{27}$$

3 METODOLOGIA

Para o desenvolvimento deste trabalho, será aplicada a metodologia ilustrada no fluxograma da Figura 10, desde a extração e segmentação dos dados, até a modelagem matemática, obtenção de resultados e discussão.

Figura 10 – Metodologia



Fonte: o autor (2021)

3.1 AMBIENTE DE APLICAÇÃO

Os algoritmos citados neste trabalho foram aplicados por meio da linguagem *Python*. Essa é uma linguagem de programação interpretada, orientada a objetos e de alto nível, com semântica dinâmica. A sintaxe simples e fácil de aprender do *Python* enfatiza a legibilidade e, portanto, reduz o custo de manutenção do programa. Além disso, ela suporta módulos e pacotes, o que incentiva a modularidade do programa e a reutilização de código (ROSSUM, 1998). O interpretador utilizado foi o Google Colab.

As análises e manipulações na base de dados foram possíveis por meio do uso da biblioteca *Pandas*. Segundo Wes McKinney (2010), ela é um pacote *Python* que fornece estruturas de dados rápidas, flexíveis e expressivas, projetadas para facilitar o trabalho com dados estruturados (tabulares, multidimensionais, potencialmente heterogêneos).

Algumas operações matemáticas, utilizando vetores e matrizes, foram realizadas por meio da biblioteca *NumPy*. De acordo Wes McKinney (2010), ela usa a sintaxe e a semântica do *Python*, operando com matrizes e empregando operações lógicas. Além disso, permite diversas abordagens orientadas a objetos e operações com tabelas usando vetores.

Os algoritmos SVM e RNA empregados neste trabalho foram implementados utilizando a biblioteca *Scikit-Learn*. Ela fornece muitos algoritmos de aprendizado não supervisionado e supervisionado, facilitando o desenvolvimento de trabalhos na área de *machine learning* (PEDREGOSA *et al*., 2011). As funcionalidades fornecidas pela Scikit-Learn incluem:

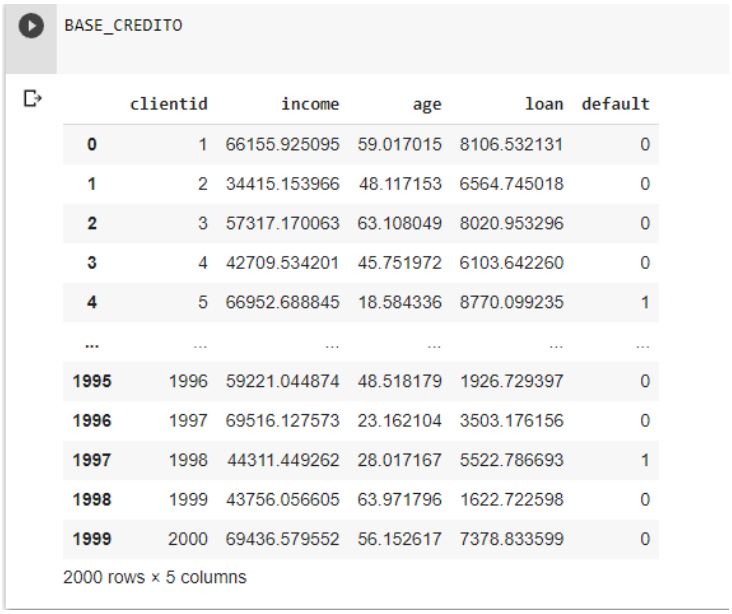
- Pré-processamento, incluindo normalização dos dados.
- Algoritmos de (RNA, SVM, KNN etc.).
- Seleção de modelo (curvas de aprendizado, validação cruzada etc.).

3.2 BASE DE DADOS

A base de dados utilizada para a aplicação dos algoritmos foi obtida no site da *Kaggle,* que mantém atualmente inúmeros conjuntos de dados gratuitamente disponibilizados para a comunidade de aprendizagem de máquina. O *dataset* em

questão contém 2.000 registros de clientes e suas características financeiras, como demonstrado na Figura 11.

Figura 11 – Base de Dados



BASE_CREDITO

	clientid	income	age	loan	default
0	1	66155.925095	59.017015	8106.532131	0
1	2	34415.153966	48.117153	6564.745018	0
2	3	57317.170063	63.108049	8020.953296	0
3	4	42709.534201	45.751972	6103.642260	0
4	5	66952.688845	18.584336	8770.099235	1
...	...	...	...	...	...
1995	1996	59221.044874	48.518179	1926.729397	0
1996	1997	69516.127573	23.162104	3503.176156	0
1997	1998	44311.449262	28.017167	5522.786693	1
1998	1999	43756.056605	63.971796	1622.722598	0
1999	2000	69436.579552	56.152617	7378.833599	0

2000 rows × 5 columns

Fonte: o autor (2021)

3.2.1 Seleção das variáveis

A base original possui 11 variáveis preditoras e 1 variável preditiva. Para a implementação do modelo, foram selecionadas 3 variáveis preditoras: *Income* (Renda anual), *Age (*Idade) e *Loan (*Dívida atual). E uma variável preditiva denominada Default, a qual só pode ter dois resultados. Quando o cliente deixou de pagar suas dívidas, a variável default é preenchida com 1, quando o cliente honrou suas obrigações, neste caso a variável default é preenchida com 0.

3.2.2 Tratamento dos dados inconsistentes

Por meio do comando *"isnull ()",* como ilustrado na Figura 12, identificou-se que a variável *Age* tinha linhas vazias, o que poderia levar o algoritmo a enviesar o resultado.

Figura 12 – Identificação De Dados Vazios

```
[ ]  BASE_CREDITO.isnull().sum()

clientid    0
income      0
age         3
loan        0
default     0
dtype: int64
```

Fonte: o autor (2021)

Para esses casos, os valores foram substituídos pelo valor médio da idade da amostra, como evidenciado na Figura 13.

Figura 13 – Retirando Campos Vazios

```
[ ]  BASE_CREDITO['age'].fillna(BASE_CREDITO['age'].mean(), inplace = True)

[ ]  BASE_CREDITO.loc[pd.isnull(BASE_CREDITO['age'])]

     clientid  income  age  loan  default
```

Fonte: o autor (2021)

Além disso, identificou-se que a variável *age* possuía algumas linhas com valores negativos, como demonstrado na Figura 14.

Figura 14 – Identificando Dados com *Age* Negativo

```
[ ]  BASE_CREDITO[BASE_CREDITO['age']<0]
```

	clientid	income	age	loan	default
15	16	50501.726689	-28.218361	3977.287432	0
21	22	32197.620701	-52.423280	4244.057136	0
26	27	63287.038908	-36.496976	9595.286289	0

Fonte: o autor (2021).

Para correção, atribuiu-se a essas linhas a idade média da amostra, que foi previamente calculada e vale 40.92 como demonstrado nas figuras a seguir.

Figura 15 – Encontrando a Média da Base

```
[8]  BASE_CREDITO['age'][BASE_CREDITO['age']>0].mean()

     40.92770044906149
```

Fonte: o autor (2021)

Figura 16 – Atribuindo o Valor Da Média

```
[ ]  BASE_CREDITO.loc[BASE_CREDITO['age']< 0, 'age'] = 40.92
```

Fonte: o autor (2021)

Após a seleção e correção das bases, as variáveis preditoras foram padronizadas para adequá-las em uma mesma ordem de grandeza, com o objetivo de evitar que o algoritmo fique enviesado para as variáveis com maiores proporções. Nessa etapa, utilizou-se o comando "*StandardScaler ()",* como demonstrado na Figura 17.

Figura 17 – Padronização das Variáveis

```
from sklearn.preprocessing import StandardScaler
scaler_credito = StandardScaler()
x_credito = scaler_credito.fit_transform(x_credito)
```

Fonte: o autor (2021)

3.3 APLICAÇÃO DA REDE NEURAL ARTIFICIAL

Para aplicação desse método, é necessário definir a topologia da rede (arquitetura, algoritmo de aprendizado e funções de ativação). Para o presente trabalho, a arquitetura de rede selecionada foi a de redes alimentadas diretamente *feedfoward* com várias camadas, ou seja, a *Perceptron* Multicamadas. A principal característica da rede é utilização de uma ou mais camadas adicionais, o que torna a rede bastante eficiente. Neste estudo, testou-se arquiteturas com 1 camada oculta (quanto mais camadas, maior o tempo de processamento), com o número de neurônios variando de 1 a 9 acrescentando 3 neurônios a cada teste. Já a quantidade de neurônios na camada de entrada é igual ao número de parâmetros preditores

inseridos na rede que são 3 (renda anual, idade e dívida atual). A quantidade de neurônios na camada de saída é sempre igual a 1, pois esse foi o único parâmetro de saída utilizado (default) que pode assumir o valor 1 quando o tomador do crédito deixou de pagar e 0 quando quitou seus débitos. Testou-se duas funções de ativação para neurônios das camadas ocultas: tangente hiperbólica (tan) e linear retificada (ReLU). Já na camada de saída foi utilizada a função de ativação sigmoide, uma vez que essa função é eficiente em casos de não linearidade, e sua maior vantagem reside no fato de que o valor da derivada é máximo quando x está próximo de 0, o que tende a "empurrar" seu resultado para as extremidades do intervalo [0, 1] ao longo do treinamento, uma característica desejável, por exemplo, em problemas de classificação de clientes. O algoritmo de aprendizado utilizado foi o *Backpropagation* de gradiente decrescente e o número máximo de iterações (épocas) foi 1000 execuções. Por fim, a taxa de aprendizagem (alpha) utilizada foi a padrão da biblioteca 0,00010 e a tolerância do erro foi de 0,00001. Na Tabela 1 está ilustrada todas as arquiteturas utilizadas.

Tabela 1 – Arquiteturas Utilizadas Na Rnas

Número da arquitetura testada	Número de neurônios na camada oculta	Função de Ativação da camada oculta
1	1	Relu
2	1	Tan
3	3	Relu
4	3	Tan
5	6	Relu
6	6	Tan
7	9	Relu
8	9	Tan

Fonte: o autor (2021)

3.4 APLICAÇÃO SVM

Para avaliação e comparação da eficiência das Máquinas de Vetores de Suporte frente às Redes Neurais Artificiais, utilizou-se o mesmo conjunto de dados. Vale salientar que para a aplicação da SVM utilizou-se a mesma biblioteca utilizada para as RNA's disponível no ambiente *python*, na qual buscou-se a avaliar quais

funções Kernel tornava o modelo mais eficiente: testou-se as funções RBF, Polinomial e Sigmoide. O componente C foi testado com o objetivo de modificar o peso de quantas amostras dentro da margem contribuem para o erro geral do algoritmo, conforme ilustrado na Tabela 2. Consequentemente, espera-se para o modelo que, com um C mais baixo, as amostras dentro das margens serão penalizadas menos do que com uma C mais alto.

Tabela 2 – Arquiteturas Utilizadas no Svms

Número da arquitetura testada	**C (parâmetro de margem)**	**Função Kernel**
1	1	Kernel Polinomial
2	1	Kernel Sigmoide
3	1	Kernel RBF
4	1,5	Kernel Polinomial
5	1,5	Kernel Sigmoide
6	1,5	Kernel RBF
7	2	Kernel RBF
8	2	Kernel Polinomial
9	2	Kernel Sigmoide

Fonte: o autor (2021)

3.5 MÉTRICAS DE VALIDAÇÃO

Na validação cruzada utilizando *k-folds*, o conjunto de dados é dividido em k segmentos. Dessa forma, um classificador é treinado usando $k-1$, partes da amostra, e um valor de acurácia (ACC) é calculado testando o classificador no segmento restante, repetindo o procedimento até que todas as amostras sejam usadas como teste, vide Figura 18.

Finalmente, estima-se a acurácia média dentre as k iterações.

Figura 18 – Validação Cruzada para 5 *Folds*

Fonte: o autor (2021)

4 RESULTADOS

Para treinar e avaliar cada algoritmo, utilizou-se o método *k-fold* para um $k = 10$. Ou seja, para cada iteração de k, a base de dados foi fracionada em 10 amostras aleatórias com 200 observações, na qual utilizou-se 9 dessas amostras para treinamento e 1 para teste, como demonstrado na Figura 19. Entretanto, contou-se com o processo de *k-fold* 10 vezes, obtendo, dessa forma, 100 iterações com 200 previsões cada. Logo, para a avaliação do desempenho de cada modelo, foi realizado a média da acurácia obtida 20.000 previsões (Figura 19).

Figura 19 – Recorte do Código da Aplicação das Rna's Com *K-Fold*

```
from sklearn.model_selection import cross_val_score, KFold

for i in range(10):
  print("i", i)
  kfold = KFold(n_splits=10, shuffle=True, random_state=i)

  rede_neural_credito1 = MLPClassifier(max_iter=1000, verbose = False, tol=0.0000100,
                                       solver = 'sgd', activation = 'relu',
                                       hidden_layer_sizes =(1))
  scores1 = cross_val_score(rede_neural_credito1, x_credit, y_credit, cv = kfold)
  resultados_rede_neural_1.append(scores1.mean())

for j in range(10):
  print("j",j)
  kfold = KFold(n_splits=10, shuffle=True, random_state=j)

  rede_neural_credito2 = MLPClassifier(max_iter=1000, verbose = False, tol=0.0000100,
                                       solver = 'sgd', activation ='tanh',
                                       hidden_layer_sizes =(1))
  scores2 = cross_val_score(rede_neural_credito2, x_credit, y_credit, cv = kfold)
  resultados_rede_neural_2.append(scores2.mean())

for k in range(10):
  print("k",k)
  kfold = KFold(n_splits=10, shuffle=True, random_state=k)

  rede_neural_credito3 = MLPClassifier(max_iter=1000, verbose = False, tol=0.0000100,
                                       solver = 'sgd', activation ='relu',
                                       hidden_layer_sizes =(3))
  scores3 = cross_val_score(rede_neural_credito3, x_credit, y_credit, cv = kfold)
  resultados_rede_neural_3.append(scores3.mean())
```

Fonte: o autor (2021)

4.1 EXPERIMENTOS COM FUNÇÃO DE ATIVAÇÃO RELU

Cada linha da Figura 20 representa a média da acurácia obtida em 200 previsões para cada subconjunto segmentado no *k-fold*. Ou seja, na linha 0 tem-se para cada arquitetura sugerida a média da acurácia para os 10 subconjuntos treinados e testados. Como foi executado 10 *k-folds*, na linha 1 tem-se a segunda média da acurácia obtida para o segundo *k-folds* executado, e assim por diante.

Figura 20 – Todos os *K-Folds* Executados

	1 NEURONIOS FUNCAO RELU	3 NEURONIOS FUNCAO RELU	6 NEURONIOS FUNCAO RELU	9 NEURONIOS FUNCAO RELU
0	0.9355	0.9590	0.9680	0.9730
1	0.9110	0.9615	0.9670	0.9695
2	0.9080	0.9645	0.9690	0.9775
3	0.9385	0.9535	0.9715	0.9735
4	0.9175	0.9585	0.9755	0.9750
5	0.8885	0.9610	0.9740	0.9740
6	0.9285	0.9500	0.9640	0.9755
7	0.9215	0.9525	0.9690	0.9690
8	0.9345	0.9570	0.9665	0.9720
9	0.9070	0.9665	0.9725	0.9660

Fonte: o autor (2021)

Com o intuito de observar a eficiência dos modelos para cada arquitetura, foram calculadas as médias das acurácias. A Tabela 3 ilustra o resultado desses cálculos. É possível aferir que houve um aumento significativo da acurácia, à medida que se aumenta o número de neurônios de 1 para 3. Para números maiores que 3, o aumento na acurácia é menor.

Tabela 3 – Resultados para Rede com Função De Ativação Relu

Função de ativação RELU				
Neurônios	1	3	6	9
Média da Acurácia	0,91905	0,95840	0,96970	0,97250
Desvio Padrão	0,01581	0,00527	0,00360	0,00345
Mínimo da Acurácia	0,88850	0,95000	0,96400	0,96600
Máximo da Acurácia	0,93850	0,96650	0,97550	0,97750

Fonte: o autor (2021)

4.1.1 Experimentos com função de ativação Tan

O mesmo processo de embaralhamento dos dados *(k-fold)* foi executado para o treinamento das RNAs com função de ativação tangente.

Analisando a Tabela 4, os resultados desta se mostraram superiores em relação aos da Tabela 3. As acurácias dos modelos com essa função de ativação são superiores em aproximadamente 2%. Nesse sentido, também é possível observar que os outros indicadores estatísticos tiveram um resultado superior em relação a mesma arquitetura da sessão anterior.

Tabela 4 – Resultados para Rede com Função De Ativação Tan

Função de ativação TAN				
Neurônios na camada oculta	1	3	6	9
Média da Acurácia	0,94335	0,95390	0,99690	0,99690
Desvio Padrão	0,00421	0,00338	0,00061	0,00039
Mínimo da Acurácia	0,93500	0,94800	0,99600	0,99650
Máximo da Acurácia	0,94750	0,96050	0,99800	0,99750

Fonte: o autor (2021)

4.2 EXPERIMENTOS COM MÁQUINA DE VETORES DE SUPORTE

Nos testes para SVM também foi utilizado o método *k-fold* com o objetivo de minimizar as chances de *overfitting*. Os algoritmos foram executados variando o parâmetro das funções Kernel: RBF, Polinomial e Sigmoide e o valor da constante C.

4.2.1 Experimentos com C igual a 1

Nesse experimento, constatou-se que o algoritmo que utilizou a função Kernel Polinomial obteve a melhor média na acurácia para o conjunto de dados analisados, como ilustrado na Tabela 5.

Tabela 5 – SVM Com Parâmetro C = 1

C=1	Kernel Polinomial	Kernel RBF	Kernel Sigmoide
Média da acurácia	0,98260	0,84490	0,96070
Desvio Padrão	0,00150	0,00077	0,00108
Mínimo da Acurácia	0,98000	0,84350	0,95950
Máximo da Acurácia	0,98500	0,84600	0,96300

Fonte: o autor (2021)

4.2.2 Experimentos com C igual a 1,5

Para o experimento com a constante C igual a 1,5 constatou-se que o algoritmo que utilizou a função Kernel Polinomial também obteve a melhor média na acurácia.

Tabela 6 – SVM Com Parâmetro C = 1,5

C=2	Kernel Polinomial	Kernel RBF	Kernel Sigmoide
Média da acurácia	0,98380	0,84335	0,96240
Desvio Padrão	0,00094	0,00154	0,00124
Mínimo da Acurácia	0,98200	0,84150	0,96050
Máximo da Acurácia	0,98550	0,84550	0,96450

Fonte: o autor (2021)

4.2.3 Experimentos com C igual a 2

Já para o experimento com a constante C igual a 2, observou-se que o algoritmo que utilizou a função Kernel Polinomial também obteve a melhor média na acurácia de todos os experimentos realizados.

Tabela 7 – SVM com Parâmetro C = 2

C=2	Kernel Polinomial	Kernel RBF	Kernel Sigmoide
Média da Acurácia	0,98490	0,84310	0,96255
Desvio Padrão	0,00080	0,00102	0,00132
Mínimo da Acurácia	0,98400	0,84150	0,96050
Máximo da Acurácia	0,98650	0,84450	0,96500

Fonte: o autor (2021)

4.3 COMPARAÇÃO ENTRE OS MODELOS

Com o intuito de definir o algoritmo com a melhor performance nas previsões realizadas, foram calculadas as acurácias médias para cada modelo e arquitetura testada nos experimentos com RNA e SVM. Analisou-se os resultados do melhor modelo para cada algoritmo. Perante isso, foi possível notar uma clara tendência de melhora nos resultados quando aumentamos o número de neurônios na camada oculta das Redes Neurais Artificiais. Constatou-se também que as redes com função de ativação tangente alcançaram os melhores resultados. Os algoritmos SVM obtiveram desempenhos muito semelhantes e uma pequena melhora à medida que o valor da constante C cresceu. Dessa forma, pode-se extrair dos experimentos que a melhor arquitetura encontrada para a RNA foi a que utilizou 9 neurônios na camada oculta e com função de ativação tangente. Já para os algoritmos SVM, o melhor modelo encontrado foi o que utilizou a função Kernel polinomial com constante C igual a 2.

5 CONSIDERAÇÕES FINAIS

O desenvolvimento deste trabalho resultou na exposição de 2 algoritmos de aprendizado de máquina destinados à predição de clientes que irão ou não cumprir com suas obrigações financeiras. Além disso, diferentes arquiteturas e parâmetros foram introduzidos. O trabalho discutiu as vantagens e desvantagens dos modelos de classificação utilizados. Os resultados encontrados demonstram que as Redes Neurais Artificias superaram as Máquinas de Vetores de Suporte em aproximadamente 1% na classificação dos clientes. A capacidade de acurácia de 99,69% demonstrou o grande poder de classificação das RNA's, o que pode auxiliar na tomada de decisão dos melhores clientes para comercializar os produtos de crédito, minimizando o risco de perdas financeiras.

Para trabalhos futuros sugere-se testar novos algoritmos de *machine learning*, como as Redes convulsionais e modelos híbridos que combinem esses algoritmos citados. Inclusive, há a possibilidade de incluir outras características financeiras e pessoais para tornar a análise mais abrangente, como nível de escolarização, profissão, tempo de trabalho remunerado, taxa de juros etc.

REFERÊNCIAS

ABE, S.; **Support Vector Machines for Pattern Classifications**. Kobe, Japão: Ed. Springer, 2005.

AZEVEDO, F. M.; BRASIL, L. M.; OLIVEIRA, R. C. L. de. **Redes neurais com aplicações em controle e em sistemas especialistas**. Florianópolis: Visual Books Editora, 2000.

BARROS, M. **Processos estocásticos**. Rio de Janeiro: Papel virtual, 2003. 432 p.

BARONE, D. **Sociedades artificiais:** a nova fronteira da inteligência das maquinas. Porto Alegre: Bookman, 2003.

BALLOU, R. H. **Gerenciamento da cadeia de suprimentos**: planejamento, organização e logística empresarial. 5. ed. Porto Alegre: Bookman, 2006.

Burman, P.: **A comparative study of ordinary cross-validation, v-fold cross-validation and the repeated learning-testing methods**. Biometrika 76, 503–514 1989.

Borra e Ciaccio, Simone Borra e Agostino Di Ciaccio. **Measuring the prediction error. a comparison of cross-validation, bootstrap and covariance penalty methods**. Computational Statistics and Data Analysis, 54:2976–2989 2010.

BRAGA, A. P.; CARVALHO, A. P. L. F.; LUDERMIR, T. B. **Redes neurais artificiais: teoria e aplicações**. Rio de Janeiro: LTC, 2000.

CHATFIELD, C. **Model Uncertainty and Forecast Accuracy**. In.: Journal of Forecasting, v. 15, p. 495-508, 1996.

CORRAR, L. J.; PAULO, E.; DIAS FILHO, J. M. **Análise multivariada para os cursos de administração, ciências contábeis e economia**. [*S. l*] Ed. Atlas, 2009.

DEMUTH, H., BEALE, M. **Neural network toolbox for use with Matlab**. ed.4. 2000. [citado em 12 de novembro de 2001]. Disponibilidade e acesso: http://www.mathworks.com.

EMC Digital Universe with Research & Analysis by IDC, 2013. **The digital universe of opportunities**: Rich Data and the Increasing Value of the Internet of Things. Acessado em: 20/11/2021. Disponível em: Acesso em: 20 nov. 2021.

FAYYAD, U. M.; PIATETSKY, S. G.; SMYTH, P.; Uthurusamy, R. "**Advances in Knowledge Discovery and Data Mining**" 1996, AAAIPress, The Mit Press.

FERNANDES, L. G.; PORTUGAL, M. S. **Redes Neurais Artificiais e Previsão de Séries Econômicas:** Uma Introdução, Textos para discussão n.95/1, Curso de Pós-Graduação em Economia, UFRGS 1995.

FIGUEIREDO, K. F.; ZACHARIAS, M. L. B.; ALMEIDA, V. M. C. de. Determinantes da Satisfação dos Clientes com Serviços Bancários. **RAE –** eletrônica, [*s. l.*], v. 7, n. 2, jul./dez. 2008.

FREEMAN, J. A.; SKAPURA, D. M. **Neural Networks Algorithms, Applications and Programmming Techniques**, [*S. l.*]: [*s. n.*], 1992.

GONÇALVES, André R. **Máquina de Vetores Suporte**, Unicamp, Campinas, Brasil 2010.

GRANGER, C. W. J.; NEWBOLD, P. **Forecasting Economic Time Series**. New York: Academic Press, 1977.

HAIR JR., J. F.; BABIN, B. J.; BLACK, W. C. **Análise Multivariada de Dados**. 5. ed. Porto Alegre: Bookman, 2005.

HANKE, J. E.; REITSCH, A. G.; WICHERN, D. W. **Business Forecasting**. 7. ed. New Jersey: Prentice Hall, 2001.

HAYKIN, S. **Redes Neurais**: Princípios e práticas. 2. ed. Porto Alegre: Ed. Artmed S. A., 1994. 903 p.

HAYKIN, S. **Redes neurais:** princípios e prática. Porto Alegre: Bookman, 2001.

HAMEL, L. (2009). **Knowledge discovery with support vector machines**. New York, NY: Wiley-Interscience.

INTERNATIONAL DATA CORPORATION. **Worldwide Big Data Technology and Services** 2012–2015 Forecast. 2013.

JORDAN, M. I., & Mitchell, T. M. (2015). **Machine learning: Trends, perspectives, and prospects**. Science, 349(6245), 255-260.

KOHAVI, R. **A study of cross-validation and bootstrap for accuracy estimation and model selection**. International Joint Conference an Artificial Intelligence (IJCAI), Stanford, CA, USA, 1995. Citado na página 67.

KOWALCZYK, A. **Support Vector Machines Succinctly**. Syncfusion, Morrisville, 2017.

LEE, Y. S.; TONG, L. I. Forecasting time series using a methodology based on autoregressive integrated moving average and genetic programming. **Knowledge-Based Systems**, [*s. l.*], n. 24, p. 66-72, 2011.

LORENA, A. C.; CARVALHO, A. C. de. **Uma introdução às support vector machines**. Revista de Informática Teórica e Aplicada, v. 14, n. 2, p. 43–67, 2007.

MARTINELI, E. **Extração de conhecimento de Redes Neurais Artificiais**. Dissertação (Mestrado em Ciências) – USP, São Paulo, 1999.

MAKRIDAKIS, S. G.; WHEELWRIGHT, S. C.; HYNDMAN, R. J. **Forecasting:** methods and applications. 3. ed. New York: John Willey & Sons, 1998.

McCULLOCH, W.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **Bulletin of Mathematical Biophysics**, v.5, p.115-133, 1943.

MEDINA, J. R.; GONZALEZ-ESCRIVA, J. A.; GARRIDO, J.; DE ROUCK, J. Overtopping analysis using neural networks. In: PROCEEDINGS OF THE 28TH INTERNATIONAL CONFERENCE ON COASTAL ENGINEERING, 28, 2002, Cardiff. **Proceedings of the**… Cardiff: ASCE, 2002. p. 2165-2177.

MUELLER, A. **Uma Aplicação de Redes Neurais Artificiais na Previsão do Mercado Acionário**. 1996. Dissertação (Mestrado em Engenharia de Produção) – Departamento de Pós-Graduação em Engenharia de Produção, Universidade Federal de Santa Catarina, Florianópolis, 1996.

NETO; MENDONÇA. **Fractais e redes neurais artificiais aplicados à previsão de retorno de ativos financeiros brasileiros**. São Paulo, [*s. n*.], 2014.

PELLEGRINI, F. R; FOGLIATTO, F. S. **Passos para implementação de sistemas de previsões de demanda: técnicas e estudo de caso**. Revista Produção, v. 11, n. 1, p. 43-64, 2001.

PEDREGOSA, F. et al. **Scikit-learn: Machine learning in Python**. Journal of Machine Learning Research, v. 12, p. 2825Ű2830, 2011.

PORTUGAL, M. S. Neural networks versus time series methods: a forecasting exercise. **Revista Brasileira de Economia**, [*s. l*.], v. 49, n. 4, p. 611-629, 1995.

QU, Y. (2008). **Macroeconomic factors and probability of default. European Journal of Economics**, Finance and Administrative Sciences, 13, 192–215.

REBELO, L. D. T., **Avaliação Automática do Resultado Estético do Tratamento Conservador do Cancro de Mama**. Faculdade de Engenharia da Universidade do Porto, Porto, Portugal, 2008
ROSSUM, G. van. **What is Python? Executive Summary**. 1998.

ROCHET. (2007). ROCHET, J. **Why Are there So Many Banking Crisis**. New Jersey, USA: Princeton University Press, 2007.

SCHÖLKOPF et al., B., J.Platt, et al. (1999b). **Estimating the support of a high-dimensional distribution**. Technical Report Microsof Research 99-87, 1999.

TRIPPI, R. R.; TURBAN, E. **Neural Networks in Finance and Investing**: using artificial intelligence to improve real world performace. Salem: Probus Publishing Company, 1993.

VAPNIK, V. N. **The Nature of Statistical Learning Theory**, 1995.

WHEELWRIGHT, S. Manufacturing Strategy: Defining the Missing Link. **Strategic Management Journal**, [*s. l.*], v. 5, n. 1, p. 77-91, 1985.

ZURADA, J. M. **Intruducion to Artificial Neural Systems**. Boston, Mass: West Publishing Company, 1992.