**Lista de abreviaturas**

AVC Acidente Vascular Cerebral

TC Tomografia computadorizada

CNN Rede Neural Convolucional

FN Falso negativo

TN Verdadeiro negativo

FP Falso positivo

TP Verdadeiro positivo

FPR Taxa de falso positivo

TPR Taxa de verdadeiro positivo

ROC Característica de Operação do Receptor

AUC Área sob da curva

**Sumário**

**1 INTRODUÇÃO**

**2 FUNDAMENTAÇÃO TEÓRICA**

**2.1 Acidente vascular cerebral – AVC**

2.1.1 Fatores de risco da doença

2.1.2 Diagnóstico com tomografias computadorizadas (TC)

**2.2 Aprendizado de Máquina**

2.2.1 Aprendizado Supervisionado

2.2.1.1 Regressão Logística

2.2.1.2 Floresta Aleatória

**2.3 Aprendizagem profunda**

2.3.1 Neurônios artificiais

2.3.2 Redes Neurais Artificiais

2.3.3 Rede Neural Convolucional

**2.5 Métricas para avaliação de desempenho**

2.5.1 Acurácia

2.5.2 Matriz de confusão

2.5.3 Taxa de FP, taxa de TP, Curva ROC e AUC ROC

**3 METODOLOGIA**

**3.1 Base de dados para os fatores de risco**

**3.1.1 Variáveis categóricas Binárias: dois possíveis valores apenas**

**3.1.2 Variáveis categóricas não binárias: vários possíveis valores**

**3.1.3 Variáveis de valor contínuo**

**3.1.4 Variável ordinal**

**3.2 Base de dados para as imagens de tomografia computadorizada**

**3.3 Ferramentas utilizadas**

3.3.1 Python

3.3.2 Bibliotecas da linguagem utilizadas

**4 DESENVOLVIMENTO**

**4.1 Classificador para fatores de risco**

4.1.1 Pré-processamento - preparação geral dos dados

4.1.2 Modelagem: criação do classificador

**4.2 Rede Neural classificadora para imagens de tomografia computadorizada**

4.2.1 Correção da rotulação dos dados

4.2.2 Criação dos conjuntos de treino e de teste

4.2.3 Aumentando os dados do conjunto de dados original

4.2.4 Transferência de aprendizagem – VGG-16

4.2.5 Treinamento da rede estruturada

**5 RESULTADOS**

**5.1 Resultados dos modelos criados para classificação de fatores de risco**

**5.2 Resultados da rede neural criada para classificação de imagens de tomografia computadorizada**

**1 INTRODUÇÃO**

**2 FUNDAMENTAÇÃO TEÓRICA**

**2.1 Acidente vascular cerebral – AVC**

O Acidente Vascular Cerebral (AVC) está entre as doenças de maior destaque que existem, sendo uma das principais causas de morte, incapacitação física e internação em todo o mundo. Tal doença é caracterizada pela alteração do fluxo sanguíneo que ocorre na região cerebral, impedindo que o sangue, e consequentemente o oxigênio, chegue às células da região, causando a morte das mesmas. A morte dessas células danifica o funcionamento do cérebro e pode deixar graves sequelas nas vítimas da doença.

A alteração do fluxo de sangue do cérebro pode se originar da obstrução total ou parcial de alguma artéria da região, o que é conhecido como acidente vascular cerebral isquêmico, ou simplesmente infarto cerebral. Esse tipo de AVC é o mais comum entre os casos da doença e pode ocorrer devido a um trombo (caso de trombose) ou êmbolo (caso de embolia) presente na vítima.

Uma outra origem da alteração do fluxo de sangue mencionada é o caso do rompimento de determinado vaso sanguíneo do cérebro, o que causa uma hemorragia na região. Esse caso caracteriza o acidente vascular cerebral hemorrágico. Tal rompimento de determinado vaso altera o nível da pressão intracraniana e pode dificultar também a chegada de sangue em outras áreas não afetadas. Embora seja menos comum, esse tipo de AVC é o mais grave, tendo maiores índices de mortalidade quando comparado ao outro tipo abordado (AVC isquêmico).

Com relação ao sintomas presentes nas vítimas, dor de cabeça forte sem causa aparente; fraqueza ou formigamento da face, braço ou perna (especialmente em apenas um dos lados do corpo); alteração da fala e/ou da compreensão; alteração da visão de um ou ambos os olhos; e dificuldade, ou incapacidade, de se movimentar são sintomas bastante comuns em ambos os tipos da doença.

Em específico para o caso de acidente vascular cerebral isquêmico, tem-se de forma comum os sintomas de tontura e perda de equilíbrio ou coordenação. Já para o caso de AVC hemorrágico pode ocorrer também náusea, vômito, confusão mental e, em alguns casos, perda de consciência. Em tal caso também podem aparecer os sintomas de sonolência exagerada, alterações na frequência cardíaca e respiratória, e até mesmo convulsões. Vale ressaltar que é de grande importância que um indivíduo com os sintomas apresentados procure assistência médica o mais rápido possível. Dessa forma, a doença pode ser diagnosticada e tratada rapidamente, possivelmente resultando em maiores chances de sobrevivência da vítima e menores sequelas.

2.1.1 Fatores de risco da doença

**Existem alguns fatores que aumentam a probabilidade da ocorrência do acidente vascular cerebral, facilitando o desencadeamento da doença. Estes são denominados fatores de risco e podem ser inerentes à vida humana, maus hábitos, estilo de vida inadequado ou até mesmo questões genéticas. Os principais fatores são:**

* **Hipertensão**
* **Diabetes**
* **Obesidade**
* **Tabagismo**
* **Consumo excessivo e frequente de álcool e drogas**
* **Estresse**
* **Idade avançada (envelhecimento)**
* **Histórico familiar**
* **Sexo masculino**
* **Colesterol elevado**
* **Doenças cardiovasculares (principalmente as que produzem arritmia cardíaca)**
* **Sedentarismo**
* **Doenças do sangue**

2.1.2 Diagnóstico com tomografias computadorizadas (TC)

Além da análise de fatores de risco e sintomas, outra forma de identificar o quadro de acidente vascular cerebral em vítimas da doença é através da análise de tomografias computadorizadas. Os quadros de AVC podem apresentar características aparentes quando analisados em tais tomografias, permitindo a identificação da doença e o nível de ocorrência da mesma em determinado cérebro.

Áreas hipodensas (mais escuras) que impedem a visualização de determinada região possivelmente indicando perda de tecido total ou parcial naquele local, por exemplo, é uma característica comum nos quadros da doença. A tomografia computadorizada vem sendo utilizada como principal forma de diagnosticar quadros de AVC para definir um possível tratamento. As figuras abaixo são exemplos de cada caso utilizado neste trabalho, sendo uma para caso de AVC hemorrágico, uma para isquêmico e uma para caso de normalidade (sem presença da doença):

Figura 1 – Exemplo de imagens de tomografia computadorizada para tipos diferentes de acidente vascular cerebral.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Condição normal | AVC Hemorrágico | AVC Isquêmico |

**Fonte:**

**2.2 Aprendizado de Máquina**

O Aprendizado de Máquina, ou Aprendizagem de Máquina, é uma área da inteligência artificial que lida com o a criação de programas “inteligentes”, capazes de aprender determinados conceitos sem serem explicitamente programados para isso, capazes de adquirir conhecimento, de aprender e tomar decisões a partir desse aprendizado. Tal área explora também a capacidade de programas e sistemas conseguirem realizar previsões sobre determinadas situações, aprendendo através da análise de dados já existentes sobre tais situações.

Envolvendo conceitos estatísticos, reconhecimento de padrões e programação, a área de Aprendizado de Máquina possibilita que resultados sejam especulados de acordo com acontecimentos passados semelhantes, através de um processamento dos dados usando o reconhecimento de padrões. **Um sistema de aprendizado é um programa de computador que toma decisões baseadas em experiências acumuladas através da solução bem sucedida de problemas anteriores [1].**

Com relação à criação de uma solução utilizando conceitos de Aprendizado de Máquina, o processo geralmente envolve duas etapas, o treinamento e a avaliação. A etapa de treinamento é quando um determinado conjunto de dados é passado para o modelo processá-lo, aprendendo a reconhecer padrões e também a fazer previsões. A etapa de avaliação consiste em passar dados semelhantes ao determinado conjunto de dados utilizado no treino para que o desempenho do modelo seja medido, analisando os resultados esperados com os resultados obtidos.

No campo do Aprendizado de Máquina, existem três tipos diferentes de aprendizagem: aprendizado supervisionado, aprendizado não supervisionado e aprendizado por reforço. Os algoritmos de Aprendizado de Máquina utilizados neste trabalho pertencem à área de aprendizado supervisionado e, portanto, apenas esta será abordada com mais detalhes.

2.2.1 Aprendizado Supervisionado

O Aprendizado supervisionado utiliza dados rotulados no treinamento do modelo, o qual prevê uma**variável dependente** a partir de uma ou mais **variáveis independentes**. Existem diversos algoritmos desse tipo de aprendizado, sendo os principais: regressão linear, regressão logística, máquina de suporte vetorial, árvores de decisão, k-vizinhos mais próximos, floresta aleatória, entre outros. Os utilizados neste trabalho para desenvolver o classificador de fatores de risco foram apenas a Regressão Logística e o Classificador de Floresta Aleatória.

2.2.1.1 Regressão Logística

A Regressão Logística é um modelo estatístico capaz de modelar determinada probabilidade de um evento acontecer a partir da combinação linear de variáveis independentes entre si associadas a tal evento. Muito utilizado em situações de classificação binária, esse modelo combina variáveis independentes com respectivos pesos atribuídos a elas de modo a obter um valor a partir deste processo. Tal valor é passado para uma função logística que resulta na probabilidade de algum evento acontecer. No caso de uma classificação, tal resultado indica a probabilidade dos valores analisados pertencerem a determinada classe do problema em questão.

De forma simplificada, o algoritmo de Regressão Logística – especificamente para casos binários (utilizado neste trabalho) – ajusta uma curva, no caso usando uma função logística, que separa de forma eficiente os dados em duas classes distintas. Tal curva ajustada é obtida, de forma simplificada, através da junção das seguintes equações:

(equação I)

(equação II)

Cada variável independente passada para o modelo é interpretada como e cada peso respectivamente associado é interpretado como (*i* varia de 0 a *n*, sendo este o número total de variáveis passadas para o modelo). Através da combinação linear dos pontos , é obtido o valor *z* (equação I) e este é passado para uma função logística (equação II), tendo como resultado disso um valor (comumente determinado de ) entre 0 e 1. De forma padrão para casos binários, se for maior ou igual a 0.5, então pode-se concluir que a entrada analisada pertence à classe 01, senão, tem-se o caso sendo pertencente a outra classe da classificação binária (por exemplo, classe 02).

Figura 02 – Gráfico de uma função logística f(x) sendo utilizada para divisão de duas classes a partir do valor 0.5 (limiar entre as classes)



Fonte: Elaborado pelo autor.

Cada peso da equação é identificado como e é gerado, inicialmente, de forma aleatória, sendo ajustado durante a fase de treinamento (ajuste) do modelo. Tal fase ocorre de forma iterativa e utiliza um conjunto de valores de dimensão *m* x *n*, em que *m* é o número total de registros diferentes do conjunto de dados e *n* é o número total de variáveis de cada registro. Para cada iteração *j* (*j* varia de 0 a *m*), os valores são alterados para atingirem valores mais otimizados (que geram menores erros) tanto para casos de classe 01 quanto para casos de classe 02.

Erros extremamente baixos não necessariamente compõem um modelo eficiente. A medida que um classificador é composto por pesos muito ajustados (com erro muito baixo), ele fica exageradamente ajustado e, assim, com baixa capacidade de generalização. Esse caso é conhecido como sobreajuste e torna o modelo muito eficiente para classificar dados muito semelhantes com os que foi treinado, porém classificando de forma equivocada dados não semelhantes, o que se tem geralmente em uma situação real.

2.2.1.2 Floresta Aleatória

Floresta Aleatória é um modelo de aprendizado de máquina composto por um conjunto aleatório de Árvores de Decisão que pode ser usado em situações de regressão ou classificação, por exemplo. Especificamente para problemas de classificação, como o abordado neste trabalho, o modelo classifica de acordo com a classificação feita pelas árvores que o compõem. Dessa forma, se há um resultado classificativo mais frequente, isto é, se a maioria das árvores gerou determinada classificação, então esta será a que o modelo de floresta aleatória irá resultar em sua conclusão.

A estrutura de uma Árvore de Decisão é constituída por vários blocos de decisão comumente denominados de nós. Cada nó funciona como um bloco condicional que encaminha a entrada recebida para um de seus filhos de acordo com o valor o qual ele analisa. Tal processo é feito para todo nó pertencente à árvore até chegar em determinado nó folha que, por sua vez, representa uma das possíveis classes do problema. Portanto, em uma árvore de decisão, uma entrada percorre um dos diferentes caminhos possíveis até chegar em um nó folha que a classifica como uma das possíveis rotulações da situação. Tal caminho acabada sendo definido de acordo com os valores que pertencem a entrada em questão. A figura 03, localizada mais adiante, representa um exemplo que ilustra uma possível estrutura de uma Árvore de Decisão.

Para cada tipo de situação abordado, isto é, para cada problema e conjunto de dados, os caminhos, os nós e os critérios de divisão de cada nó mudam. A etapa de treinamento do modelo irá analisar quais divisões são mais eficientes, utilizando critérios como entropia, por exemplo, e assim definirá os melhores caminhos a serem construídos. Portanto, existem infinitas possibilidades de estruturas para árvores de decisão, sendo cada uma delas definidas de acordo com o conjunto de dados utilizado na etapa de treino do modelo.

A ideia do algoritmo de Floresta Aleatória é justamente utilizar dessas diversas possibilidades de estrutura das Árvores de Decisão e criar assim uma floresta com várias árvores diferentes abordando o mesmo problema, classificando determinada entrada recebida de acordo com a classificação mais frequente das árvores que compõem o modelo.

Figura 3 – Exemplo de estrutura de uma Árvore de Decisão para um conjunto de dados que contenha as variáveis *Idade*, *Problema* *Cardíaco*, *Alcoolismo*, *Gênero* e *Tabagismo*, com as possíveis rotulações de classe *não* *AVC* e *AVC*.



Fonte: Elaborado pelo autor.

Figura 4 – Estrutura de um modelo de Floresta Aleatória para m variáveis, contendo n Árvores de Decisão com duas possibilidades de classificação: Classe 01 e Classe 02.

****

Fonte: Elaborado pelo autor.

**2.3 Aprendizagem profunda**

2.3.1 Neurônios artificiais

O neurônio artificial é o principal componente das redes neurais computacionais e seu funcionamento é baseado nos neurônios reais, que compõem o sistema nervoso. Os neurônios reais, base do sistema nervoso, são células que estabelecem conexões entre si para transmitir impulsos nervosos pela região cerebral. O neurônio artificial é uma estrutura com funcionamento que se assemelha do neurônio real, mantendo a principal característica dessas estruturas de criarem conexões entre si, recebendo informações provenientes de outros neurônios e passando elas adiante.

Os principais componentes de um neurônio artificial são as entradas, os pesos, a função de ativação e saída. As entradas, geralmente representadas pela letra *x*, recebem a informação que chega para o neurônio. Tal informação é multiplicada por um determinado peso, comumente representado pela letra *w*, e o resultado disso é passado para uma função de ativação que processa os valores recebidos e retorna um ou mais valores, os quais servirão de entrada para um outro neurônio artificial conectado à rede.

Outro componente muito comum nos neurônios das redes neurais artificiais é o viés (bias). Tal valor é processado junto com os dados *x\*w* na função de ativação como tentativa de tornar o neurônio não muito tendencioso, isto é, não muito apegado aos dados que recebe. Isso é de grande utilidade para que a rede neural possa ficar não enviesada, tendo um melhor desempenho com relação a resultados corretos quando em contato com dados não conhecidos, graças à capacidade de generalização obtida nesse processo.

Figura 5 – Ilustração de uma possível estrutura para um neurônio artificial



Fonte: Elaborado pelo autor.

2.3.2 Redes Neurais Artificiais

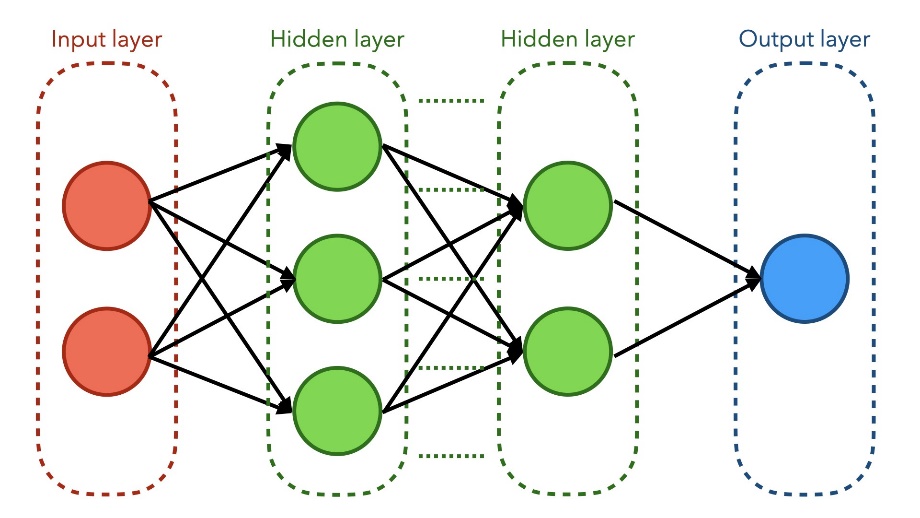
Redes neurais artificiais são estruturas do campo da inteligência artificial que possibilitam os computadores a processar dados se baseando no comportamento do cérebro humano. Ao receber um conjunto de dados específico, uma rede neural artificial é capaz de se ajustar a este conjunto, aprendendo a reconhecer padrões sobre estes dados e se tornando mais eficiente à medida que os processa. A estrutura da rede é composta por um conjunto de neurônios artificiais agrupados em camadas interconectadas que possibilitam o computador a aprender, reconhecer padrões e tomar decisões inteligentes.

Uma rede neural simples possui camadas de três tipos: camada de entrada, camada oculta e camada de saída. A camada de entrada é onde se encontram os neurônios responsáveis por receber os dados que serão processados pela rede. Tal camada faz o primeiro processamento desses dados e os encaminham para o próximo conjunto de neurônios. As camadas do tipo oculta recebem dados da camada de entrada ou de outras camadas ocultas, processando novamente os dados e passando o resultado para outra camada. Redes neurais podem ter uma ou mais camadas desse tipo.

Por fim, o conjunto de neurônios da camada de saída faz o último processamento dos dados, fornecendo o resultado final de todos os dados processados pela rede. Para problemas de classificação binária, tem-se apenas um nó na camada de saída, o qual indica a probabilidade de determinada entrada analisada pela rede ser ou não da classe abordada. Já para problemas de várias classes diferentes, tem-se vários nós nessa camada.

Além dessa estrutura de rede abordada, existem outros tipos mais específicos de redes neurais que apresentam melhores resultados quando aplicados em determinadas situações. No caso de problemas que envolvem a classificação de imagens e o reconhecimento de objetos, o uso de Redes Neurais Convolucionais é mais recorrente do que o uso da rede neural padrão, justamente pela boa capacidade que estas possuem de valorizar características da imagem.

Figura 6 – Ilustração simplificada de uma Rede Neural Artificial simples



Fonte: [An Illustrated Guide to Artificial Neural Networks | by Fahmi Nurfikri | Towards Data Science](https://towardsdatascience.com/an-illustrated-guide-to-artificial-neural-networks-f149a549ba74)

2.3.3 Redes Neurais Convolucionais

Comumente chamadas de Conv-Net ou simplesmente CNN, as redes neurais do tipo convolucional são muito utilizadas em tarefas que envolvem a identificação de objetos, reconhecimento de características e classificação de imagens. Esse tipo de rede tem a capacidade de valorizar detalhes da imagem que são relevantes para a classificação, realçando determinados agrupamentos de pixels através da aplicação de filtros matriciais na imagem, o que colabora para um processo de classificação mais eficiente.

As Redes Neurais Convolucionais apresentam três tipos principais de camadas em sua estrutura: camadas de convolução, camadas de agrupamento (*pooling*) e as camadas totalmente conectadas. Camadas de convolução, ou convolucionais, são as camadas responsáveis pelo processo de realce das características mencionado no parágrafo anterior.

Esse processo se dá pela aplicação de um filtro, comumente chamado de kernel, na imagem, vista como uma matriz de pixels, que está sendo processada pela rede, funcionando como um detector de recursos. Esse filtro tem formato matricial com tamanho típico de 3x3, podendo ser maior dependendo da rede, e o processo de aplicação se dá de forma iterativa via a realização do produto escalar entre seus valores e os da matriz da imagem processada, finalizando-se com a soma dos resultados dessa multiplicação.

Tal processo é iterativo, acontecendo até que o kernel multiplique toda a imagem. Alguns hiperparâmetros que podem variar o volume da imagem resultante são: o número de filtros que serão aplicados a imagem, o número de passos denominado *stride* que o kernel percorre para se deslocar de uma multiplicação para a outra e o *padding*, que adiciona zeros na matriz da imagem para chegar a determinado tamanho que favoreça a aplicação do kernel.

As camadas de agrupamento irão pegar o resultado da camada de convolução e reduzir o número de valores recebidos, mantendo as características realçadas pelos filtros da convolução, porém eliminando áreas vazias de informação relevante para a classificação. O processo de *pooling* destas camadas se dá por dois tipos diferentes que podem ser utilizados, o agrupamento máximo ou o agrupamento médio.

O agrupamento máximo é feito através da aplicação de um filtro que se desloca pela imagem que simplesmente pega o maior valor dentre os valores analisados na imagem e o coloca em uma matriz de saída, descartando os demais valores. O agrupamento médio se dá pela aplicação de um filtro que também se move pela imagem analisada, porém calculando o valor médio dos valores analisados pelo filtro, colocando-o em uma matriz de saída de forma semelhante ao outro agrupamento descrito. O agrupamento máximo tem maior frequência de utilização quando comparado com o agrupamento médio.

Por fim, nas camadas totalmente conectadas tem-se diversos neurônios totalmente conectados que irão processar os valores recebidos aplicando alguma função de ativação, gerando valores de saída para as próximas camadas ou para a camada de saída. Toda rede neural convolucional tem seu formato base composto por essas camadas, porém existem diversos tipos de CNN diferentes que vão variar o número de instâncias dessas camadas mencionadas e os hiperparâmetros presentes nas mesmas.

A arquitetura de rede convolucional denominada de VGG-16, utilizada no desenvolvimento deste trabalho, apresenta treze camadas de convolução e cinco camadas de agrupamento, assim como três camadas totalmente conectadas. A imagem abaixo ilustra a distribuição dessas camadas, com as camadas de agrupamento entre grupos de camadas de convolução e as camadas totalmente conectadas localizadas no final da rede:

Figura 7 – Possível representação de uma rede da arquitetura de Rede Neural Convolucional VGG-16



Fonte: [Fig. A1. The standard VGG-16 network architecture as proposed in [32].... | Download Scientific Diagram (researchgate.net)](https://www.researchgate.net/figure/Fig-A1-The-standard-VGG-16-network-architecture-as-proposed-in-32-Note-that-only_fig3_322512435)

**2.5 Métricas para avaliação de desempenho**

Com o modelo classificador ajustado de acordo com a situação abordada, torna-se fundamental a realização de uma boa avaliação de seu desempenho. **O objetivo da fase de avaliação é estimar os resultados de mineração de dados de forma rigorosa e obter a confiança de que são válidos e confiáveis antes de avançar [2].** Avaliar de forma adequada permite a identificação de problemas, como sobreajuste ou excesso de generalização, por exemplo, assim como visualizar de forma geral como seria a performance do modelo quando colocado em produção na prática.

Para realizar tal processo de avaliação nos modelos criados neste trabalho, foram definidas algumas métricas utilizadas na avaliação de modelos classificadores, as quais serão abordadas logo abaixo.

2.5.1 Acurácia

A acurácia é uma das métricas mais simples de se medir e justamente por isso sua utilização é muito popular na avaliação de classificadores. Porém, por ser demasiadamente simples, tal métrica pode esconder deficiências do modelo analisado e transparecer um resultado não muito confiável. O cálculo da acurácia é dado pela divisão do número de acertos do modelo pelo número total de casos analisados, ou seja, pelo número de acertos somados com o número de erros:

Como o cálculo dessa métrica não leva em conta fatores além de acertos e erros, como o tipo de erro, por exemplo, é bastante interessante utilizar a Acurácia em conjunto com outras métricas para fazer uma avalição mais eficiente, levando em conta também os tipos de valores falsos que o modelo gera.

2.5.2 Matriz de confusão

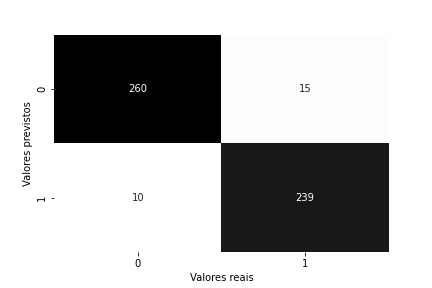
Na fase de avaliação de um modelo também pode ser interessante olhar para o desempenho do modelo com relação a erros específicos. Analisar a taxa de falso positivo, erro do tipo I, e de falso negativo, erro do tipo II, permite visualizar melhor qual tipo de erro é predominante na questão, assim como ter uma ideia percentual probabilística de como o modelo em questão reagiria em situações reais com relação aos tipos de erros.

Além dos tipos de erros (falsos), tem-se também dois diferentes casos de acertos, sendo estes os verdadeiros positivos e verdadeiros negativos. É comumente utilizado para simplificar a nomenclatura dessas métricas as seguintes siglas: FN para falso negativo, TP para verdadeiro positivo, TN para verdadeiro negativo e FP para falso positivo.

A matriz de confusão é a métrica que permite a análise desses valores de falsos e verdadeiros de uma forma simplificada. Ela é bastante utilizada na avaliação de classificadores pois fornece uma visão sobre a confusão entre as classes que um determinado classificar realizou na análise de casos passados para ele. Uma matriz de confusão é uma matriz de dimensão *n* x *n* sendo que *n* é o número de classes do problema abordado.

**Uma matriz de confusão separa as decisões tomadas pelo classificador, tornando explícito como uma classe está sendo confundida com outra. Desta forma, diferentes tipos de erros podem ser tratados separadamente [2]**. Na matriz de confusão temos dois tipos de acertos e erros, os verdadeiros positivos (TP) e negativos (TN), e os falsos positivos (FP) e negativos (FN), respectivamente. Valores verdadeiros ocorrem quando o modelo classifica corretamente valores negativos (0) ou positivos (1) e valores falsos ocorrem quando se tem a classificação incorreta desses mesmos valores, negativos ou positivos.

Figura 8 – Exemplo de matriz de confusão para um classificador binário em que as classes são rotuladas como 0 (negativo) e 1 (positivo). No eixo vertical tem-se a rotulação 0 ou 1 para os valores previstos pelo modelo e no eixo horizontal, a rotulação dos valores reais



Fonte: Elaborado pelo autor.

2.5.3 Taxa de FP, taxa de TP, Curva ROC e AUC ROC

A taxa de FP, ou FPR, mede a fração dos valores negativos que foram previstos incorretamente, julgados como positivos pelo classificador em questão. Para realizar o cálculo dessa métrica, a seguinte equação é utilizada:

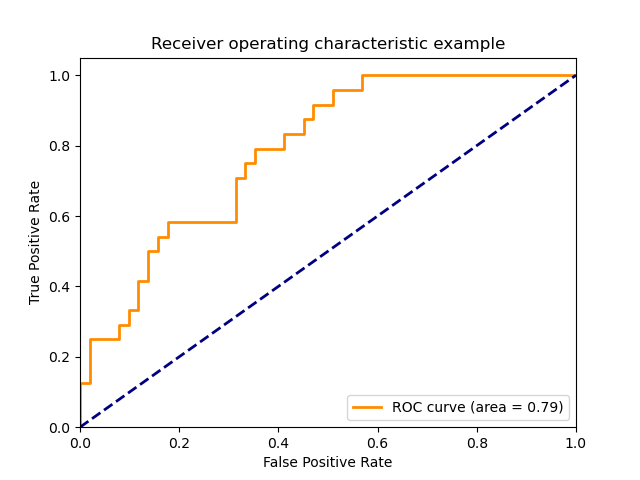
A taxa de TP, ou TPR, também conhecida como sensibilidade, mede a fração dos valores positivos que foram previstos de forma correta, julgados também como positivos pelo modelo. O cálculo da TPR é feito da seguinte forma:

É de grande importância que sejam analisadas ambas as métricas descritas acima em conjunto, pois assim torna-se possível ter um diagnóstico confiável e saber se o modelo está extremamente tendencioso ou não. Por exemplo, em uma situação em que determinado classificador rotula sempre todos os dados como sendo de uma classe positiva, sua TPR terá valor máximo (1.0), pois para todo valor positivo recebido, ele acertará todos. Porém, este mesmo modelo, como consequência, errará todos os casos negativos, e isso só poderá ser enxergado se a FPR for analisada, pois seu valor também será 1.0.

Nesse cenário descrito, analisar apenas a TPR indicaria um modelo muito eficiente, já que ele não errou nenhum caso positivo que recebeu. Dessa forma, estaria sendo omitido o alto viés e a baixa capacidade de generalização do classificador em questão. Um modelo tendencioso assim possivelmente apresentaria péssimos resultados na prática.

Para facilitar a análise em conjunto de ambas as métricas, existe uma outra métrica muito poderosa denominada de curva ROC (Característica de Operação do Receptor, em português). Ela é uma ferramenta gráfica que permite verificar de forma conjunta TPR e FPR à medida que o limiar entre as classes (probabilidade que separa as classes) do modelo varia. Tal variação é feita para diferentes valores do limiar que geram diferentes valores de TPR e FPR como resultado. Estes valores formam pontos em um gráfico TPR x FPR e a junção desses pontos formam a curva ROC.

Figura 9 – Visualização gráfica de uma curva ROC (em laranja) constituída por pontos obtidos a partir de diferentes valores para o limiar entre as classes de determinado modelo preditivo. É possível visualizar também no gráfico uma curva *x = y* (em azul) de referência que indica pontos de TPR e FPR de valores iguais.



Fonte: [3.3. Métricas e pontuação: quantificando a qualidade das previsões — documentação do scikit-learn 1.1.3](https://scikit-learn.org/stable/modules/model_evaluation.html#roc-metrics)

Outra forma de utilizar a curva ROC para avaliação de determinado modelo é calculando sua área. A partir deste processo, tem-se um número que pode variar entre 0 a 1.0, sendo que, quanto mais próximo de 1.0, mais eficiente o modelo seria, com máximo TPR e mínimo FPR. A área abaixo da curva ROC é uma métrica denominada de Pontuação AUC (*area under curve*) ROC.

**3. METODOLOGIA**

A metodologia deste trabalho se consistiu no primeiro momento a entender mais sobre os assuntos abordados, através de um levantamento bibliográfico sobre os temas de Aprendizado de Máquina e Aprendizado Profundo, estudando os principais modelos e técnicas dessas áreas para criar uma solução correta e eficiente do problema abordado. Para ambos os modelos criados aqui, foram seguidas as etapas: levantamento e pré-processamento, aplicação de técnicas das áreas, dos dados que envolvem o problema (fatores de risco e imagens de tomografia computadorizada), processamento desses dados para criação do modelo classificador e fase final de avaliação do mesmo, para entender o desempenho do modelo criado.

3.1 Base de dados para os fatores de risco

Os dados utilizados na construção do modelo de Aprendizado de Máquina para fatores de risco foram retirados da plataforma *Kaggle*, sendo um dos conjuntos de dados disponibilizados para propósitos educacionais. O conjunto original conta ao todo com 5110 registros, em que cada um contém informações de uma pessoa diferente, rotulada como vítima de AVC ou não. O conjunto original estava desbalanceado, com a grande maioria dos valores sendo de pessoas rotuladas como não AVC. As características presentes no conjunto de dados são:

**3.1.1 Variáveis categóricas Binárias: dois possíveis valores apenas**

* **Sexo: Masculino ou feminino, em que o sexo masculino é um fator de risco**
* **Hipertensão: Indivíduo é ou não hipertenso. Hipertensão é um fator de risco.**
* **Doença cardíaca: Indivíduo tem ou não alguma doença cardíaca, o que é um fator de risco.**
* **Estado civil de ser ou já ter sido casado: Sim ou não, tal variável pode estar ligada a estilo de vida e estresse, o qual é um fator de risco da doença.**
* **Tipo de residência: Urbana ou rural, tal variável está ligada a estilo de vida, que pode ser um fator considerável para a doença**
* **Vítima de AVC: Sim ou não, essa é a variável alvo do trabalho, sendo utilizada para rotular os dados para previsão.**

**3.1.2 Variáveis categóricas não binárias: vários possíveis valores**

* **Tipo de emprego: empregado privado, funcionário público (governo), empreendedor ou não trabalha.**
* **Status com relação a tabagismo: indivíduo nunca fumou, fuma formalmente, fuma regularmente ou situação desconhecida.**

**3.1.3 Variáveis de valor contínuo:**

* **Idade: idade avançada é um fator de risco para a doença**
* **Nível de glicose: diabetes é um fator de risco para a doença**
* **BMI: Índice de massa corpórea no padrão americano, utilizado para identificar obesidade.**

**3.1.4 Variável ordinal:**

* **Id: Identificação de cada registro. Tal variável foi desconsiderada por não ter impacto na variável alvo do trabalho que rotula os casos de AVC.**

3.2 Base de dados para as imagens de tomografia computadorizada

Para criação do modelo classificador de imagens de TC, foi utilizado um conjunto de dados com 475 imagens no total, divididas em três classes diferentes: AVC Isquêmico, AVC Hemorrágico ou não AVC. Entretanto, como o objetivo deste trabalho é de apenas identificar a ocorrência de Acidente Vascular Cerebral em determinada vítima, sem especificar o tipo de AVC de fato, este conjunto foi redistribuído para apenas duas classes.

Dessa forma, o conjunto original passou a ter apenas as classes AVC e não AVC, com as seguintes distribuições:

* AVC possui 301 imagens
* Não AVC possui 174 imagens

3.3 Ferramentas utilizadas

3.3.1 Python

Python é uma linguagem de programação de alto nível, gratuita e muito utilizada no desenvolvimento de aplicações modernas. Criada por volta de 1990, tal linguagem apresenta um uso mais simples quando comparado com o uso de outras linguagens do mercado, trazendo mais facilidade para quem não é necessariamente do ramo da programação. Ela prioriza a legibilidade do código, combinando uma sintaxe clara com recursos poderosos de sua biblioteca padrão (recursos nativos) e de outras bibliotecas desenvolvidas pela comunidade.

Por ser acessível e amigável, a linguagem de programação Python se tornou bastante popular no mercado, com muitos frameworks e módulos prontos desenvolvidos e disponibilizados para os usuários da ferramenta. Bibliotecas de áreas como Ciência de Dados, Automação, Aprendizado de Máquina, Aprendizado profundo, Redes Neurais, são bastante famosas e utilizadas pelos usuários da linguagem, podendo ser facilmente importadas para que não haja a necessidade de implementar funções complexas e componentes desde o início.

A alta disponibilidade de bibliotecas e recursos para as áreas de Aprendizado de Máquina e Aprendizado profundo, assim como a facilidade de uso da linguagem foram os principais critérios para escolha do uso dessa linguagem para o desenvolvimento deste trabalho. Abaixo estão listadas as principais bibliotecas Python criadas por terceiros e utilizadas para o desenvolvimento da solução proposta.

3.3.2 Bibliotecas da linguagem utilizadas

* *Numpy*: utilizada para trabalhar com vetores e matrizes de forma eficiente e facilitada, possibilitando operações matemáticas velozes com essas estruturas em grandes dimensões e complexidades. Otimizada com a linguagem de programação C, Numpy traz velocidade e performance bastante agradável para essas operações no Python.
* *Pandas:* biblioteca Python para análise e manipulação de dados. Permite gerenciar dados em grandes volumes, aplicando operações matemáticas de diferentes graus de complexidade de uma forma bastante flexível, fácil, rápida e poderosa. Diversas operações presentes nesta biblioteca são construídas em cima de componentes *Numpy*.
* *Matplotlib*: biblioteca base para muitas bibliotecas de visualização de dados gráfica, compreende a criação de gráficos estáticos, animados e interativos em Python.
* *Seaborn:* biblioteca Python para visualização de dados em alto nível. Baseada na biblioteca Matplotlib, Seaborn traz muitas implementações de gráficos prontas como: gráfico de linhas, gráfico de barras, gráfico de pontos de calor, etc.
* *Scikit-learn*: também conhecida como Sk Learn, é uma das principais ferramentas para se trabalhar com Aprendizado de Máquina na linguagem Python. Tal biblioteca traz diversas implementações de modelos de regressão, classificação e agrupamento, assim como a implementação de várias métricas de avaliação bastante conhecidas como: acurácia, precisão, matriz de confusão, erro quadrado médio, curva ROC, etc. Dessa forma, o programador que utiliza o Sk Learn pode se preocupar com o uso dos modelos e seus resultados mais do que com a criação do modelo em si.

Construída a partir da junção de outras bibliotecas como Numpy, Matplotlib e SciPy (biblioteca que implementa métodos estatísticos em Python), a Scikit-learn traz consigo também diversos componentes para otimização/ajuste de hiperparâmetros, possibilitando que modelos eficientes sejam criados e aperfeiçoados para contextos mais específicos.

Sk Learn traz também algumas implementações de componentes para divisão de dados inteligente de forma mais inteligente e elaborada, como algoritmos para validação cruzada de dados, e algoritmos para normalização dos dados, como normalização padrão, normalização min-max, entre outras. Com essa biblioteca também é possível gerar \textit{datasets} e aplicar técnicas para correção de conjuntos já existentes, por meio de técnicas como a OneHotEncoder, por exemplo, a qual é facilmente aplicada com a Sk Learn.

* *Tensor Flow*: plataforma de código aberta que facilita a criação de modelos de Aprendizagem de Máquina e Aprendizagem profunda. Tensor Flow traz possibilidades para que o usuário do framework possa trabalhar com modelos prontos ou cria-los de acordo com as necessidades. Com esta biblioteca é possível facilmente estruturar redes neurais simples e complexas, alterando o número de neurônios, função de ativação, adicionando diferentes tipos de camadas, camadas de pré-processamento, entre outros.

**4. DESENVOLVIMENTO**

**4.1 Classificador para fatores de risco**

**4.1.1 Pré-processamento - preparação dos dados**

Antes de iniciar o processo de criação de um modelo preditivo, tanto para classificação quanto para regressão, é necessário preparar os dados que serão utilizados nesse processo. Isto é necessário para que o melhor desempenho deste modelo a ser criado seja alcançado.

As tecnologias analíticas que podemos utilizar são poderosas, mas impõem determinados requisitos sobre os dados que usam. Com frequência, elas exigem que os dados estejam em uma forma diferente de como são fornecidos naturalmente, e alguma conversão será necessária. (PROVOST e FAWCETT, 2016, p. 29).

Dessa forma, algumas alterações nos dados foram feitas com intuito de prepara-los para uso correto na criação do classificador. Ao analisar a distribuição da variável alvo do conjunto de dados, foi encontrado a princípio desbalanceamento dos valores. Com 4861 casos de não ocorrência de AVC e apenas 249 casos para ocorrência da doença, seria sintetizado um classificador enviesado para casos de não AVC, classificando dados que receberia de forma tendenciosa.

Para realizar o balanceamento dessas duas classes (*avc* e *não avc*) foi realizado o seguinte procedimento:

1. Separação dos elementos do conjunto original em duas classes: *avc* e *não avc*.
2. Redução aleatória do conjunto de dados da classe *não avc*: de 4861 elementos para 275.
3. Junção do conjunto de dados da classe *avc* (249 elementos) com o novo conjunto de dados reduzido da classe *não avc* (276 elementos), totalizando 525 elementos.

Portanto, com o balanceamento das classes presentes na variável alvo, tem-se um novo conjunto de dados de 525 elementos com 52.6% dos dados sendo casos de AVC e o restante para casos de condição normal (não AVC). Esse novo conjunto de dados criado foi utilizado ao decorrer de todo este trabalho para desenvolver o modelo classificador.

Com relação ao formato de cada atributo que compõe o conjunto de dados, alguns não apresentavam formato numérico e, portanto, tiveram que ser corrigidos para que não houvesse erros na criação do classificador. As variáveis contínuas presentes no conjunto de dados já estavam com o formato adequado numérico e não precisaram de correção.

Tabela 1 – Tabela com a frequência de cada possível valor dos atributos categóricos presentes no conjunto de dados

|  |  |
| --- | --- |
| Atributo | Distribuição das frequências |
| Gênero | Masculino: 41.71%  Feminino: 58.29% |
| Hipertensão | 0: 82.48%  1: 17.52% |
| Doença cardíaca | 0: 88.76%  1: 11.24% |
| É / já foi casado | Sim: 80%  Não: 20% |
| Tipo de emprego | Setor privado: 62.86%  Autônomo: 22.09%  Cargo público: 14.29%  Nunca trabalhou: 0.76% |
| Tipo de residência | Urbano: 50.86%  Rural: 49.14% |
| Tabagismo | Nunca fumou: 37.33%  Fuma formalmente: 24.57%  Fuma constantemente: 22.48%  Situação desconhecida: 15.62% |
| AVC | 0: 52.57%  1: 47.43% |

Para corrigir o formato textual de variáveis categóricas que apresentam apenas duas opções de valores, basta apenas substituir uma delas pelo dígito “1” e a outra pelo ”0”. Já para casos em que as colunas apresentam mais do que dois possíveis valores, é possível realizar a correção das mesmas através da técnica de *OneHotEncoder*. Tal transformador recebe uma coluna com diferentes possíveis valores textuais e cria novas colunas binárias para cada valor diferente.

Figura 10 - Exemplo de correção para a coluna com apenas duas opções de valores (variável binária).

|  |  |  |
| --- | --- | --- |
| Gênero |  | Masculino |
| Masculino | 1 |
| Masculino | 1 |
| Feminino | 0 |
| Feminino | 1 |
| Feminino | 0 |
| Masculino | 0 |
| Feminino | 0 |

Fonte: Elaborado pelo autor

Figura 11 - Exemplo de aplicação de OneHotEncoder para correção de coluna com quatro opções de valores diferentes.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tipo de emprego | OneHotEncoder | Privado | Autônomo | Cargo público | Criança |
| Privado | 1 | 0 | 0 | 0 |
| Privado | 1 | 0 | 0 | 0 |
| Autônomo | 0 | 1 | 0 | 0 |
| Privado | 1 | 0 | 0 | 0 |
| Criança | 0 | 0 | 0 | 1 |
| Cargo público | 0 | 0 | 1 | 0 |
| Autônomo | 0 | 1 | 0 | 0 |

Fonte: Elaborado pelo autor.

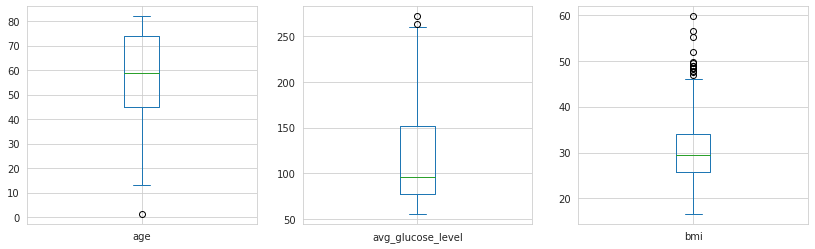
Como visto acima, toda linha obrigatoriamente contém o dígito “1” em pelo menos uma das novas colunas criadas. Isso ocorre, pois, a criação dessas colunas é feita a partir da ocorrência de todos os diferentes possíveis valores, de forma a ter uma nova coluna para cada valor diferente presente. Por essa razão, pode-se deduzir o valor de uma coluna a partir dos valores das demais, já que os valores são exclusivos.

Além do formato numérico das variáveis, também é necessário lidar com a presença de dados nulos no conjunto de dados para evitar erros durante o processo de ajuste do modelo. Dessa forma, cada variável presente no conjunto de dados foi analisada com relação a presença desse tipo de dado e a única que apresentou os dados nulos foi *BMI*, com 7.6% (40 elementos em um total de 525 registros) de valores desse tipo.

Para correção dos valores nulos encontrados em *BMI* fora calculada a mediana dos valores não nulos dessa coluna e depois feita a substituição dos valores nulos por essa mediana calculada. Após tal procedimento, todos os valores nulos do conjunto de dados foram corrigidos, já que só existiam valores assim nessa coluna.

Outro aspecto importante que pode acabar atrapalhando a criação do modelo é a presença de anomalias, ou *outliers*,no conjunto de dados. Tais valores podem atrapalhar a análise de determinado conjunto, pois são sujeiras e geralmente trazem certo ruído às medidas estatísticas da respectiva distribuição de valores que pertencem. Para identificação desse tipo de valor, foi realizada a análise gráfica das distribuições das variáveis quantitativas do conjunto de dados utilizado (Figura 12).

Figura 12 - Gráfico de distribuição dos atributos contínuos presentes no conjunto de dados com a presença de anomalias*.*



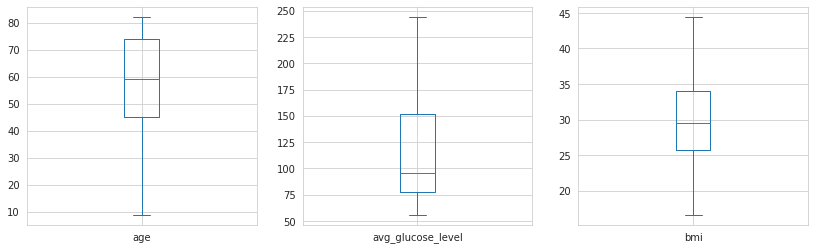
Fonte: Elaborado pelo autor.

Com a identificação das anomalias, agora pode-se iniciar a etapa de tratamento desses valores. Para isso, fora calculado o IQR dos atributos quantitativos e posteriormente foram determinados o limite superior e o limite inferior dos valores de cada atributo:

1. IQR**:** 3° quartil – 1° quartil
2. Limite superior: Mediana + 1.5 x IQR
3. Limite inferior**:** Mediana - 1.5 x IQR

Com tais medidas calculadas, tornou-se possível substituir todo valor maior que o limite superior pelo próprio limite superior e todo valor menor que o limite inferior pelo próprio limite inferior. Após a realização desses procedimentos, as anomalias foram corrigidas e o resultado obtido está expresso logo abaixo:

Figura 13 - Gráfico de distribuição dos atributos contínuos presentes no conjunto de dados sem a presença de anomalias, após as alterações realizadas nos valores*.*



Fonte: Elaborado pelo autor.

Como algumas variáveis numéricas apresentaram ainda intervalos diferentes, a normalização foi realizada para que tal fato não prejudicasse a criação do classificador. Modelos de aprendizado de máquina lineares podem acabar sofrendo com a falta de padronização dos intervalos. Uma variável que varia de 0 a 80, enquanto outra varia de 50 a 200, por exemplo, pode acabar prejudicando o modelo linear. Por essa razão, nesses casos é interessante se realizar a normalização.

A normalização é uma técnica geralmente aplicada como parte da preparação de dados para o aprendizado de máquina. O objetivo da normalização é mudar os valores das colunas numéricas no conjunto de dados para usar uma escala comum, sem distorcer as diferenças nos intervalos de valores nem perder informações. A normalização também é necessária para alguns algoritmos para modelar os dados corretamente.

A normalização evita na modelagem problemas com tais diferenças nos intervalos das variáveis. Ela mantém os valores originais, porém em uma escala padrão. Como a normalização utilizada aqui foi a de máximos e mínimos, através da função *sklearn.preprocessing.MinMaxScaler()*, o padrão de escala ficou entre 0 e 1. **TEM QUE COLOCAR RODAPÉ COM O LINK DA DOCUMENTAÇÃO DO SKLEARN SOBRE ESSA FUNÇÃO**

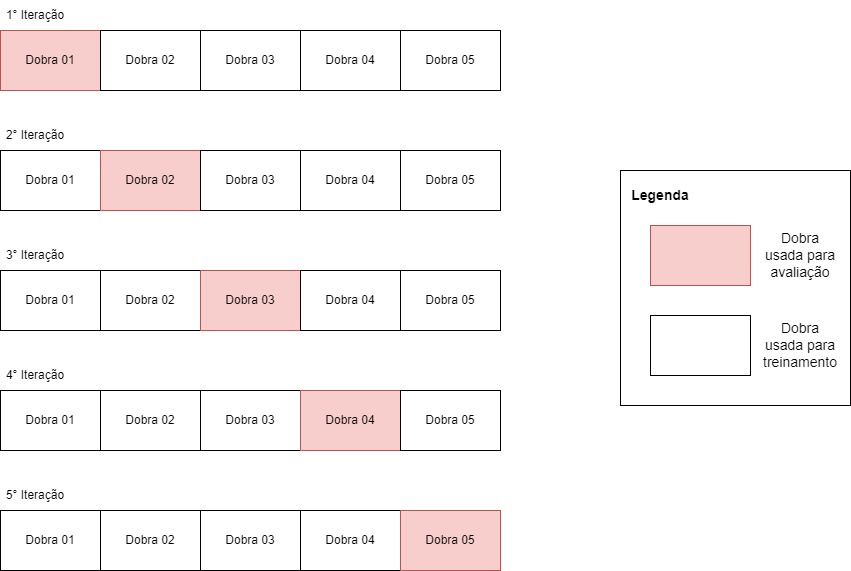
**4.1.2 Modelagem: criação do classificador**

Com a etapa de preparação dos dados feita, o conjunto de dados está pronto para ser processado. Para iniciar o processo de criação do modelo, a modelagem, o conjunto total de dados disponível foi dividido em partes distintas de treino e teste, como forma de evitar o sobreajuste do modelo criado a partir da análise de tais dados. Utilizando uma parte diferente para treino e uma diferente para teste, a avaliação do modelo é feita a partir de dados desconhecidos pelo modelo, mostrando um resultado mais verdadeiro e confiável.

Porém, apenas essa divisão pode não ser suficiente para conjuntos pequenos, pois mesmo criando os conjuntos de treino e teste de forma aleatória, pode ocorrer a seleção de valores mais representativos dependendo da semente definida para aleatoriedade. Por essa razão, uma boa maneira de lidar com conjuntos pequenos é fazer várias divisões de treino e teste no conjunto, pois assim, treina-se o modelo e o avalia com diferentes subconjuntos de dados, trazendo resultados ainda mais confiáveis. Uma forma bastante comum de se fazer isso é através do uso da Validação Cruzada e essa foi a forma utilizada aqui.

A divisão dos elementos foi feita com cinco dobras: os 525 elementos presentes no conjunto de dados total foram divididos em cinco grupos de 125 elementos cada e, então, foram separados quatro deles para treinamento do modelo e um para teste. Esse processo de separação foi repetido por cinco vezes de forma que o modelo fosse novamente treinado e testado com diversas combinações de conjuntos. Essa validação cruzada foi utilizada para criação de dois modelos classificadores diferentes, um a partir do algoritmo de Regressão Logística e outro a partir do algoritmo de Floresta Aleatória.

Figura 14 – Ilustração do processo de validação cruzada, com cinco dobras, feito para treinamento e avaliação dos dois classificadores criados. A junção das cinco dobras é o conjunto de dados total utilizado.



Fonte: Elaborado pelo autor.

O classificador com Regressão Logística foi criado a partir da classe *sklearn.linear\_model.LogisticRegression()*, com o valor 7 definido como semente para geração de valores aleatórios e os demais parâmetros com valores padrões da biblioteca. O modelo para classificação com Floresta Aleatória foi criado a partir da classe *sklearn.ensemble.RandomForestClassifier()*,com a semente para geração de valores aleatórios também sendo 7, com duzentas árvores de decisão e com os demais parâmetros com valores padrões da biblioteca.

**4.2 Rede Neural classificadora para imagens de tomografia computadorizada**

4.2.1 Correção da rotulação dos dados e criação dos conjuntos de treino e de validação

O conjunto de dados original divide as imagens de tomografia computadorizada nas categorias de condição normal, condição de avc hemorrágico e condição de avc isquêmico. Porém, para abordar de forma mais condizente com o trabalho proposto, uma nova rotulação foi feita. As imagens foram movidas para uma nova distribuição de duas classes: avc (hemorrágico e isquêmico) e não avc (condição normal).

Sobre o tamanho desse novo conjunto gerado a partir da nova rotulação das classes feita, a classe *avc* possui 299 imagens enquanto que a classe *não avc* possui 174, sendo que a dimensão de cada imagem do conjunto é de 512 x 512 pixels. A respeito da distribuição dessas imagens para treino e validação, com intuito de evitar sobre ajuste, o conjunto de imagens foi distribuído como 65% (309 imagens) dos dados para treinamento e 35% (166 imagens) para validação. Dessa forma, a rede pode aprender com os dados de treinamento e ser avaliada com os dados de validação, os quais ela ainda não conhece por não terem sido usados na etapa de treino.

Tabela 2 – Distribuição das imagens do conjunto de dados antes da nova rotulação

|  |  |  |
| --- | --- | --- |
|  | Condição | Imagens |
| Rotulação original | Normal | 174 (36.8%) |
| AVC Hemorrágico | 142 (30%) |
| AVC Isquêmico | 157 (33.2%) |

Fonte: Elaborado pelo autor.

Tabela 3 – Distribuição das imagens do conjunto de dados depois da nova rotulação

|  |  |  |
| --- | --- | --- |
|  | Condição | Imagens |
| Nova rotulação | AVC | 299 (63.2%) |
| não AVC | 174 (36.8%) |

Fonte: Elaborado pelo autor.

4.2.3 Pré-processamento: redimensionamento dos valores e aumento dos dados do conjunto de dados original

Como o conjunto de dados utilizado aqui é composto por imagens, tem-se então um agrupamento de pixels, uma matriz, para cada imagem. Um pixel possui sua coloração definida a partir de um valor, para cada canal de cor, que varia de 0 a 255, totalizando 256 elementos. Embora seja possível realizar o treinamento de uma rede com imagens compostas por pixels de valores entre 0 a 255, redimensionar esse intervalo para valores de 0 a 1 trazem melhores resultados para o modelo, pois os valores que serão multiplicados pelos pesos dos nós acaba sendo menor.

Portanto, foi realizado o redimensionamento dos dados por meio da adição de uma camada inicial na rede do tipo *Rescaling* com valor para divisão de 255. Dessa forma, toda imagem (matriz) que entra na rede já tem seus pixels redimensionados, com seus valores divididos por 255, que é o maior valor possível presente nos pixels.

Outra etapa do pré-processamento realizado foi o aumento de informação do conjunto de dados para que melhores resultados fossem obtidos. Como mencionado na seção anterior, o conjunto de dados disponível conta com 473 imagens no total. Embora seja possível realizar o treinamento de uma rede neural com esse número de registros, aumentar o número de imagens, e consequentemente aumentar a quantidade de informação passada para a rede, pode gerar resultados mais satisfatórios e uma rede neural mais eficiente.

Uma forma bastante comum no campo do Aprendizado Profundo de aumentar os dados de um conjunto, quando não se tem a possibilidade de adicionar novas imagens externas ao conjunto em questão, é utilizar técnicas de aumento dos dados (*data augmentation*), de modo que a quantidade de informação no conjunto cresça a partir das imagens já presentes no conjunto, sem que de fato sejam adicionadas novas imagens no conjunto.

Para realizar o aumento da informação do conjunto de dados foram adicionadas camadas no início da rede neural que fazem algumas alterações no conteúdo da imagem, caracterizando uma nova imagem, porém sem prejudicar as informações usadas para a classificação. Ao todo foram três camadas que aplicam inversão, rotação e ajustes de contraste nas imagens recebidas, respectivamente.

Tabela 4 – Alterações feitas para aumento dos dados do conjunto de imagens utilizado.

|  |  |  |
| --- | --- | --- |
| Alteração aplicada | Fator | Método utilizado |
| Inversão (*flip*) | Horizontal/vertical | *tensorflow.keras.layers.RandomFlip*("horizontal\_and\_vertical", seed=10) |
| Rotação | 0.2 | *tensorflow.tf.keras.layers.RandomRotation*(0.2, seed=7) |
| Contraste | Valor entre [0.3, 0.5] | *tensorflow.keras.layers.RandomContrast*(factor=(0.3, 0.5), seed=(7,17)) |

Fonte: Elaborado pelo autor.

4.2.4 Estrutura da Rede Neural Convolucional desenvolvida

Para desenvolver a solução proposta neste trabalho, foi criada uma Rede Neural Convolucional para classificação binária das imagens de TC nas classes de *avc* e *não avc*. A estrutura dessa rede conta com as camadas que implementam o pré-processamento, comentado nas seções anteriores, e com a transferência de aprendizado de uma rede VGG-16 pré-treinada, contendo as camadas de convolução e de agrupamento dessa arquitetura, com parâmetros já definidos e pesos já gerados a partir do conjunto de dados *imagenet* ([ImageNet (image-net.org)](https://www.image-net.org/)), que contém diversas imagens pré-rotuladas para treinamentos avançados de Aprendizado Profundo.

Além da VGG-16, a rede desenvolvida neste trabalho conta também com uma camada de achatamento (*flatten*), com duas camadas densas de neurônios com ativação *ReLU*, com 64 e 32 neurônios, respectivamente, e com uma última camada contendo apenas um neurônio, tendo este como função de ativação a função sigmoide, a qual gera o valor usado para classificação final de determinada imagem processada pela rede.

Tabela 5 – Estrutura final da Rede Neural Convolucional criada para imagens RGB de dimensão 512x512 pixels

|  |  |  |  |
| --- | --- | --- | --- |
|  | Componentes da rede | Principal função do componente | Tamanho da saída |
| 1 | Camada de entrada e redimensionamento | Redimensionamento de cada pixel de 0 a 255 para 0 a 1 | (512, 512, 3) |
| 2 | Camada de inversão | Inversão aleatória (horizontal/vertical) | (512, 512, 3) |
| 3 | Camada de rotação | Rotação de 0.2 com sentido aleatório | (512, 512, 3) |
| 4 | Camada de contraste | Contraste aleatório com fator de 0.3 a 0.5 | (512, 512, 3) |
| 5 | Camadas de convolução e agrupamento VGG-16 | Transferência de aprendizado de uma CNN VGG, com pesos fixos pré-gerados a partir do conjunto *imagenet* | (16, 16, 512) |
| 6 | Camada de achatamento | Redimensiona saída para apenas um valor | (131072) |
| 7 | Camada densa de neurônios | 64 neurônios com função de ativação ReLU | (64) |
| 8 | Camada densa de neurônios | 32 neurônios com função de ativação ReLU | (32) |
| 9 | Camada densa de neurônios | Apenas 1 neurônio com função de ativação sigmoide | (1) |

Fonte: Elaborado pelo autor

4.2.5 Treinamento da rede estruturada

Para realizar o treinamento da rede criada, fora feita antes a compilação do modelo, definindo a forma de calculo do erro como entropia cruzada binária e uma taxa de aprendizado de 0.001. Ademais, fora definida uma função *call-back* para interromper o treinamento caso o modelo alcançasse acurácia de validação de 95% e AUC ROC de validação de 0.9.

O treinamento foi feito a partir do conjunto de dados de treino, composto por dados desconhecidos pelo modelo, com 12 épocas (iterações que percorrem todo o conjunto de dados) e com tamanho do lote (*batch size*) de 32 imagens por iteração do treinamento de cada época. As camadas da rede pré-gerada VGG-16, integrada na rede via transferência de aprendizado, não tiveram seus pesos alterados durante o treinamento feito, pois manter os pesos originais resultou em melhores resultados em comparação com treiná-los novamente a partir do conjunto de dados utilizado neste trabalho.

**5 RESULTADOS**

O processo de avaliação é uma das principais partes de qualquer projeto que envolva a criação de modelos de preditivos. Após a criação da, é preciso analisar seu desempenho e, assim, medir sua eficiência. É de grande importância que este processo seja feito de forma correta para que os resultados da avaliação não sejam enganos e não escondam falhas do modelo.

**5.1 Resultados dos modelos criados para classificação de fatores de risco**

Como forma de medir o desempenho geral dos classificadores criados, a acurácia de cada um foi calculada a partir da função *sklearn.metrics.accuracy\_score()*, da biblioteca Scikit-Learn, para cada dobra de teste utilizada na avaliação de ambos os modelos. Como consequência, tem-se, portanto, uma sequência de cinco valores e a acurácia final foi definida como a média desses valores.

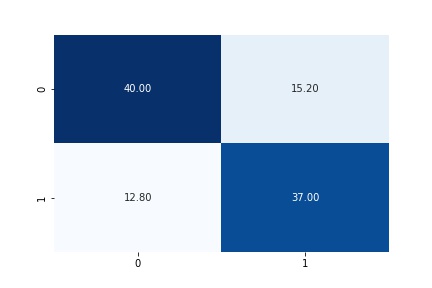
Tabela 2 – Valor médio e desvio padrão das acurácias calculadas em cada iteração de teste da validação cruzada, para ambos os classificadores criados

|  |  |  |
| --- | --- | --- |
| Acurácia do classificador | Regressão Logística | Média: 73.33%  Desvio padrão: 2.48% |
| Floresta Aleatória | Média: 73.14%  Desvio padrão: 1.40% |

Fonte: Elaborado pelo autor.

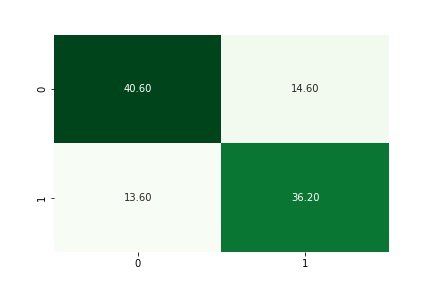
Também foi gerada e armazenada, para cada dobra de teste da validação cruzada feita para criação dos modelos, uma matriz de confusão. Após a execução de todas as iterações da validação, a média e o desvio dos valores foram calculados e os resultados, para o modelo de Regressão Logística e para o modelo de Floresta Aleatória, podem ser vistos nas figuras 15 e 16 respectivamente.

Figura 15 – Matriz de confusão para o modelo de Regressão Logística, criado a partir da média dos valores obtidos nos testes da validação cruzada.



Fonte: Elaborado pelo autor.

Figura 16 – Matriz de confusão para o modelo de Floresta Aleatória, criado a partir da média dos valores obtidos nos testes da validação cruzada.



Fonte: Elaborado pelo autor.

Além da visualização da frequência de cada tipo de erro e acerto feita pela geração das matrizes de confusão de cada modelo, foi feita também a análise das taxas de falso positivo e verdadeiro positivo para os dois modelos:

Tabela 3 – Média e desvio padrão da taxa de valores verdadeiros positivos gerados em cada iteração de teste da validação cruzada

|  |  |  |
| --- | --- | --- |
| TPR | Regressão Logística | Média: 74.53%  Desvio padrão: 74.53% |
| Floresta Aleatória | Média: 72.95%  Desvio padrão: 4.11% |

Fonte: Elaborado pelo autor.

Tabela 4 – Média e desvio padrão da taxa de valores falsos positivos gerados em cada iteração de teste da validação cruzada

|  |  |  |
| --- | --- | --- |
| FPR | Regressão Logística | Média: 27.28%  Desvio padrão: 5.27% |
| Floresta Aleatória | Média: 26.23%  Desvio padrão: 3.99% |

Fonte: Elaborado pelo autor.

Para saber sobre o desempenho dos modelos com relação a TPR e FPR a partir da variação de limiares de classificação para o modelo, ou seja, se o modelo apresenta boa capacidade de classificação com relação a dados positivos e negativos, foi calculada a área sob a curva ROC de cada modelo após a validação cruzada feita:

Tabela 5 – Média e desvio padrão dos valores de AUC ROC calculados em cada iteração de teste da validação cruzada para ambos os modelos

|  |  |  |
| --- | --- | --- |
| AUC ROC | Regressão Logística | Média: 0.8029  Desvio padrão: 0.0506 |
| Floresta Aleatória | Média: 0.7982  Desvio padrão: 0.0349 |

Fonte: Elaborado pelo autor.

**5.2 Resultados da rede neural criada para classificação de imagens de tomografia computadorizada**

Para medir a eficiência da rede neural desenvolvida, as mesmas métricas da seção anterior foram utilizadas. A acurácia atingida pelo classificador demonstrou boa capacidade de acerto geral do mesmo, com relação a ambos os conjuntos, de treino e de validação, processados.

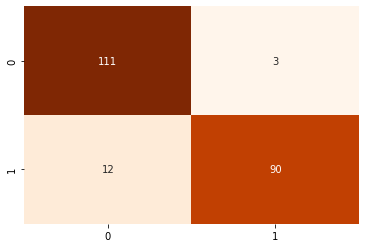
Tabela 6 – Acurácia alcançada na classificação dos dados utilizados para treinamento e validação da rede neural desenvolvida

|  |  |  |
| --- | --- | --- |
| Acurácia da Rede | Dados de treino | 93.06% |
| Dados de validação | 97.39% |

Fonte: Elaborado pelo autor.

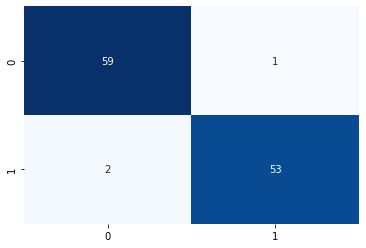
Com relação aos tipos de erros, o classificador de imagens de TC apresentou bons resultados quanto ao equilíbrio de ambos, mantendo casos de falso positivo e falso negativo com baixas frequências de ocorrência. Duas matrizes de confusão foram calculadas para análise desses valores, uma para dados de treinamento, já conhecidos pelo modelo por conta da fase de ajuste, e uma para dados de validação, com as rotulações não conhecidas pela rede neural.

Figura 17 – Matriz de confusão com os valores gerados na classificação feita pela rede a partir do conjunto de dados de treino, com valores conhecidos pelo classificador

****

Fonte: Elaborado pelo autor.

Figura 18 – Matriz de confusão com os valores gerados na classificação feita pela rede a partir do conjunto de dados de validação, com valores desconhecidos pelo classificador

****

Fonte: Elaborado pelo autor.

As taxas de verdadeiro positivo e falso positivo também foram calculadas para os dados de treinamento e para os dados de validação, com intuito de fornecer uma análise mais precisa desses casos. Os valores obtidos indicam bons resultados, com grandes taxas de acertos para casos positivos e baixas taxas de falsos positivos.

Tabela 7 – Taxa de valores verdadeiros positivos alcançada na classificação dos dados utilizados para treinamento e validação da rede neural desenvolvida

|  |  |  |
| --- | --- | --- |
| TPR | Dados de treino | 88.24% |
| Dados de validação | 96.36% |

Fonte: Elaborado pelo autor.

Tabela 8 – Taxa de valores falsos positivos alcançada na classificação dos dados utilizados para treinamento e validação da rede neural desenvolvida

|  |  |  |
| --- | --- | --- |
| FPR | Dados de treino | 2.63% |
| Dados de validação | 1.67% |

Fonte: Elaborado pelo autor.

Para analisar o desempenho da rede neural criada com relação ao cálculo de TPR e FPR a partir da variação de limiares de classificação, foi calculada a área sob a curva ROC para os dados, conhecidos, de treino e também para os dados, não conhecidos, de validação. Os resultados mostram que a rede apresenta bons acertos mesmo com a variação do limiar, indicando boa capacidade de classificação da mesma.

Tabela 9 – AUC ROC calculadas na classificação dos dados utilizados para treinamento e validação da rede neural desenvolvida

|  |  |  |
| --- | --- | --- |
| AUC ROC | Dados de treino | 0.9802 |
| Dados de validação | 0.9944 |

Fonte: Elaborado pelo autor.

**6 CONCLUSÃO**

O trabalho desenvolvido neste documento teve como principal intuito propor formas de classificar casos de AVC a partir da análise de informações, fatores de risco e imagens de tomografia computadorizada (TC), sobre vítimas da doença. Para isso, em um primeiro momento foi realizado o levantamento bibliográfico do problema, realizando estudos sobre a doença em si para saber como ela se caracteriza no indivíduo, quais são os motivos para sua ocorrência, qual o comportamento das vítimas e os principais fatores de risco englobados na doença.

Além de conhecer sobre o Acidente Vascular Cerebral, também foram realizadas pesquisas bibliográficas sobre as áreas de Aprendizagem de Máquina (\textit{Machine Learning}) e Aprendizagem Profunda (\textit{Deep Learning}), conhecendo sobre técnicas importantes dessas áreas para o desenvolvimento da solução proposta. Ademais, foram pesquisados conjuntos de dados sobre fatores de risco e imagens de TC para serem utilizados no treinamento de modelos classificadores.

Para o caso da classificação de fatores de risco, foram desenvolvidos dois modelos utilizando os classificadores de Regressão Logística e Floresta Aleatória. A avaliação de ambos os modelos gerou resultados bastante semelhantes, com acurácia de X para o modelo de Regressão Logística e Y para o de Floresta Aleatória. Com relação à ocorrência do pior erro possível da situação abordada, isto é, o caso de falso negativo, em que o modelo classifica alguma vítima como alguém de condição normal, ambos os modelos tiveram resultados próximos, porém o modelo TAL teve menor ocorrência desses erros, sendo, portanto, uma solução mais eficiente.

Já para o caso de processamento e classificação de imagens de TC, foi desenvolvida uma Rede Neural do tipo Convolucional a partir do processo de transferência de aprendizado de uma rede pré-treinada, do tipo VGG-16. Após à realização de ajustes de alguns hiperparâmetros e à adição de algumas camadas densas de neurônios, o treinamento do classificador desenvolvido gerou resultados como X de acurácia e Y para taxa de falso positivo

Referências

AVC

<https://www.gov.br/saude/pt-br/assuntos/saude-de-a-a-z/a/avc>

<https://bvsms.saude.gov.br/avc-acidente-vascular-cerebral/>

<http://departamentos.cardiol.br/dha/revista/8-3/acidente.pdf>

<https://dco-unesp-bauru.github.io/tcc-bcc-2020-2/BrunaLT/thesis-BrunaLT.pdf#page=39&zoom=100,113,261>

<https://scielo.br/j/rbepid/a/KFNpCf4NCd8mhcsT4FqJsHP/?lang=pt#:~:text=Segundo%20os%20registros%20no%20Sistema,tratamento%20do%20AVC1-4>.

<https://pebmed.com.br/como-identificar-o-avc-isquemico-na-tomografia-computadorizada/>

<https://www.medway.com.br/conteudos/como-identificar-os-sinais-precoces-do-avc-na-tc/>

ML

<https://profs.info.uaic.ro/~ciortuz/SLIDES/2017s/ml0.pdf>

[1] <https://dcm.ffclrp.usp.br/~augusto/publications/2003-sistemas-inteligentes-cap4.pdf>

<https://lamfo-unb.github.io/2017/07/27/tres-tipos-am/>

<https://www.ibm.com/topics/logistic-regression>

<https://scikit-learn.org/stable/modules/linear_model.html>

<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

<https://www.datacamp.com/tutorial/random-forests-classifier-python>

<https://en.wikipedia.org/wiki/Random_forest>

https://en.wikipedia.org/wiki/Decision\_tree

DP

<https://drauziovarella.uol.com.br/corpo-humano/neuronio/#:~:text=Compartilhar-,Neur%C3%B4nios%20s%C3%A3o%20as%20c%C3%A9lulas%20que%20caracterizam%20o%20sistema%20nervoso%2C%20respons%C3%A1veis,externo%20ou%20do%20pr%C3%B3prio%20organismo>.

<https://www.gsigma.ufsc.br/~popov/aulas/rna/neuronio_artificial/index.html#:~:text=O%20neur%C3%B4nio%20artificial%20%C3%A9%20um,Esquema%20do%20neur%C3%B4nio%20biol%C3%B3gico.&text=Viu%2Dse%20o%20c%C3%A9rebro%20como%20um%20sistema%20computacional>.

<https://aws.amazon.com/pt/what-is/neural-network/#:~:text=Uma%20rede%20neural%20%C3%A9%20um,camadas%2C%20semelhante%20ao%20c%C3%A9rebro%20humano>.

<https://towardsdatascience.com/why-we-need-bias-in-neural-networks-db8f7e07cb98>

<https://arxiv.org/abs/1511.08458>

<https://www.upgrad.com/blog/basic-cnn-architecture/#:~:text=other%20advanced%20tasks.-,What%20is%20the%20architecture%20of%20CNN%3F,the%20main%20responsibility%20for%20computation>.

<https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939>

<https://www.jeremyjordan.me/convnet-architectures/>

<https://www.ibm.com/cloud/learn/convolutional-neural-networks>

Métricas de avaliação:

Livro data Science para negócios [2]

[sklearn.metrics.accuracy\_score — scikit-learn 1.1.3 documentation](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html)

[sklearn.metrics.confusion\_matrix — scikit-learn 1.1.3 documentation](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html#:~:text=sklearn.metrics.%20confusion_matrix%20%28y_true%2C%20y_pred%2C%20%2A%2C%20labels%20%3D%20None%2C,and%20predicted%20to%20be%20in%20group%20%28j%29%20.)

[ROC Curve, a Complete Introduction | by Reza Bagheri | Towards Data Science](https://towardsdatascience.com/roc-curve-a-complete-introduction-2f2da2e0434c)

[3.3. Métricas e pontuação: quantificando a qualidade das previsões — documentação do scikit-learn 1.1.3](https://scikit-learn.org/stable/modules/model_evaluation.html#roc-metrics)

[Erro do tipo I – Wikipédia, a enciclopédia livre (wikipedia.org)](https://pt.wikipedia.org/wiki/Erro_do_tipo_I)

Metodologia – base de dados para fatores de risco

[Stroke Prediction Dataset | Kaggle](https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset)

Metodologia – base de dados para imagens

Ferramentas

Python: <https://pt.wikipedia.org/wiki/Python>

Numpy: <https://numpy.org/>

Seaborn <https://seaborn.pydata.org/>

Matplotlib <https://matplotlib.org/>

Pandas: <https://pandas.pydata.org/>

Scikit Learn: <https://scikit-learn.org/stable/>

Tensor flow: <https://www.tensorflow.org/overview>

Desenvolvimento

PROVOST, Foster; FAWCETT, Tom. Data Science para Negócios: O que Você Precisa Saber Sobre Mineração de Dados e Pensamento Analítico de Dados. Rio de Janeiro, RJ: Alta Books, 2016.

<https://docs.microsoft.com/pt-br/azure/machine-learning/component-reference/normalize-data>

<https://docs.aws.amazon.com/pt_br/machine-learning/latest/dg/cross-validation.html>

<https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html>

<https://dco-unesp-bauru.github.io/tcc-bcc-2020-2/BrunaLT/thesis-BrunaLT.pdf#page=39&zoom=100,113,261>

<https://www.aprendadatascience.com/blog/an%C3%A1lise-explorat%C3%B3ria-de-dados/vari%C3%A1veis-categ%C3%B3ricas-enconding>

<https://cloud.google.com/architecture/data-preprocessing-for-ml-with-tf-transform-pt1?hl=pt-br>

<https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset>

<https://revistas.pucsp.br/index.php/redeca/article/download/23377/16824/60265>

<https://datascience.eu/pt/aprendizado-de-maquina/validacao-cruzada-k-fold/>