UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Instituto de Ciência e Tecnologia
Câmpus de Sorocaba

PEDRO NOGUEIRA DE CARVALHO

Inteligência Artificial aplicada a um classificador de estilos estéticos na fotografia de moda

Sorocaba
2024

PEDRO NOGUEIRA DE CARVALHO

Inteligência Artificial aplicada a um classificador de estilos estéticos na fotografia de moda

Trabalho de Conclusão de Curso apresentado ao Instituto de Ciência e Tecnologia de Sorocaba, Universidade Estadual Paulista (UNESP), como parte dos requisitos para obtenção do grau de Bacharel em Engenharia de Controle e Automação.

Orientador: Prof. Dr. Leopoldo André Dutra Lusquino Filho

Sorocaba

2024

C331i

Carvalho, Pedro Nogueira de

Inteligência Artificial aplicada a um classificador de estilos estéticos na fotografia de moda / Pedro Nogueira de Carvalho. -- Sorocaba, 2024

70 p.

Trabalho de conclusão de curso (Bacharelado - Engenharia de Controle e Automação) - Universidade Estadual Paulista (UNESP), Instituto de Ciência e Tecnologia, Sorocaba

Orientador: Prof. Dr. Leopoldo André Dutra Lusquino Filho

1. Fotografia. 2. Aprendizado do computador. 3. Processamento eletrônico de dados. 4. Redes neurais (Computação). 5. Automação. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca da Universidade Estadual Paulista (UNESP), Instituto de Ciência e Tecnologia, Sorocaba. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Instituto de Ciência e Tecnologia
Câmpus de Sorocaba

INTELIGÊNCIA ARTIFICIAL APLICADA A UM CLASSIFICADOR DE ESTILOS ESTÉTICOS NA FOTOGRAFIA DE MODA

PEDRO NOGUEIRA DE CARVALHO

ESTE TRABALHO DE GRADUAÇÃO FOI JULGADO ADEQUADO COMO PARTE DOS REQUISITOS PARA A OBTENÇÃO DO GRAU DE **BACHAREL EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO**

Prof. Dr. Everson Martins
Coordenador

BANCA EXAMINADORA:

Prof. Dr. LEOPOLDO ANDRÉ DUTRA LUSQUINO FILHO
Orientador/UNESP – Campus de Sorocaba

Prof. Dr. LUIS ARMANDO DE ORO ARENAS
UNESP – Campus de Sorocaba

Prof. Dr. LUIZ FERNANDO DOS REIS DE OLIVEIRA
UFRJ - Universidade Federal do Rio de Janeiro

Junho de 2024

AGRADECIMENTOS

Expresso minha gratidão à minha família, que desde sempre tem me dado total suporte e apoio para que eu pudesse realizar meus estudos e aprendizados durante toda a graduação, e à minha noiva, Giulia Silva, que tem sido um alento e suporte moral, encorajando-me e transbordando palavras e atitudes que me fizeram seguir em frente. Agradeço aos meus colegas, cuja amizade supera os anos e ambientes universitários. Também sou grato aos professores, coordenadores, diretores e ao setor administrativo, que garantiram um ambiente que me possibilitou o acesso à minha verdadeira vocação.

Agradeço às modelos que disponibilizaram o direito de imagem para a elaboração do presente projeto. Também sou grato ao meu professor orientador, Leopoldo André Dutra Lusquino Filho, que sempre esteve de prontidão para auxiliar durante todo o processo, direcionando e esclarecendo todas as dúvidas.

Por fim, reconheço também a graça e a misericórdia que sempre estiveram disponíveis todas as manhãs por parte de Jesus Cristo, meu Senhor e Deus, que nunca me desamparou e sempre me susteve de pé.

RESUMO

O trabalho propõe utilizar técnicas de Machine Learning, especialmente Redes Neurais Convolucionais (CNNs), para classificar fotografias de moda, automatizando a seleção e organização de imagens. Destaca-se a importância da coleta meticulosa de dados e do uso de ferramentas como Keras/TensorFlow e Scikit-learn. Para realizar a classificação das imagens foram utilizadas três algorítmos: SVM, Random Forest Classifier e CNN. Afim de realizar as análises foram extraídos principalmente 4 métricas, sendo precisão, acurácia, *recall* e *F1-score*. Os resultados destacam a eficácia das CNNs na identificação de padrões visuais complexos, além da importância de uma abordagem cuidadosa na coleta e preparação dos dados para garantir resultados confiáveis e precisos.

Palavras-chave: fotografia de moda; machine learning; redes neurais convolucionais; classificação de imagens; automatização; eficiência.

ABSTRACT

This study proposes to employ Machine Learning techniques, especially Convolutional Neural Networks, to classify fashion photographs, automating image selection and organization. The importance of meticulous data collection and the use of tools like Keras/TensorFlow and Scikit-learn is emphasized. Three algorithms were used for image classification: SVM, Random Forest Classifier, and CNN. In order to conduct the analysis, four metrics were primarily extracted: precision, accuracy, recall, and F1-score. The results highlight the effectiveness of CNNs in identifying complex visual patterns, as well as the significance of a careful approach to data collection and preparation to ensure reliable and accurate results.

Keywords: fashion photography; machine learning; convolutional neural networks; image classification; automation; efficiency.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1.INTRODUÇÃO

A era digital marcou uma revolução na forma como lidamos com informações visuais, impulsionando uma inundação de fotografias e imagens em diversas plataformas online. Esse fenômeno não apenas reflete a busca incessante da humanidade por simplificar tarefas e melhorar a vida, como também ressalta a evolução contínua das tecnologias ao longo da história. Desde tempos remotos, os seres humanos têm desenvolvido ferramentas especializadas para diversas finalidades, desde a criação de roupas até a construção de máquinas sofisticadas. Segundo Ribeiro (2006), essa trajetória de inovação e a busca constante por soluções mais eficientes e avançadas nos levaram ao momento atual, onde a inteligência artificial surge como uma poderosa aliada em diversas áreas, incluindo a fotografia de moda.

No contexto atual da fotografia de moda, essa grande demanda de imagens é particularmente evidente, com profissionais buscando constantemente maneiras de otimizar seus fluxos de trabalho e atender às demandas de um mercado altamente competitivo, como elabora a Redação DC (NÚMERO..., 2021). Com a rápida evolução da tecnologia, os fotógrafos e profissionais da moda enfrentam o desafio de lidar com uma quantidade cada vez maior de imagens, tornando a organização, seleção e edição uma tarefa complexa e demorada. Nesse cenário, a inteligência artificial se destaca como uma solução promissora para lidar com esse volume de dados e otimizar os processos envolvidos na produção de fotografias de moda de alta qualidade.

O objetivo deste trabalho é explorar e desenvolver métodos de classificação de fotografias de moda baseados em machine learning (ML), uma abordagem que permite às máquinas aprenderem e executarem tarefas complexas sem a necessidade de programação explícita. Ao analisar um vasto conjunto de dados de imagens de moda, os algoritmos de aprendizado de máquina podem identificar padrões e características comuns entre as diferentes poses, expressões faciais e estilos de moda. Essa capacidade de reconhecimento automático permite uma classificação automatizada das imagens, facilitando a organização e seleção das fotografias de moda de acordo com critérios específicos, como estilo, cor, tema ou modelo.

Durante o desenvolvimento deste projeto, serão investigados os mais recentes avanços em algoritmos de Aprendizado Profundo, em particular Redes Neurais Convolucionais (CNNs), devido à sua eficácia comprovada em tarefas de visão computacional, incluindo classificação de imagens. Através da análise e experimentação com esses algoritmos, será possível avaliar o desempenho e a eficácia de diferentes abordagens na classificação de fotografias de moda.

Além disso, aspectos relacionados à coleta e pré-processamento de dados específicos da fotografia de moda serão discutidos, destacando a importância de garantir a qualidade e a representatividade do conjunto de dados utilizado no treinamento do modelo de classificação. Os direitos das fotografias do banco de dados ficam reservados devido à importância de proteger a imagem e a identidade das modelos, bem como de garantir o cumprimento das regulamentações de direitos autorais e privacidade de dados.

Em um cenário onde a tecnologia desempenha um papel cada vez mais relevante, torna-se fundamental investigar e estudar o potencial das I.As em reconhecer e classificar imagens. Cada nova aplicação desenvolvida testa o potencial dos algoritmos para lidar com diferentes dados e novas tarefas, demonstrando a necessidade contínua de pesquisa e avaliação do desempenho da inteligência artificial em diversas áreas e com diferentes tipos de dados. Este trabalho visa contribuir para o avanço do conhecimento na área de classificação de fotografias de moda, oferecendo insights sobre as melhores práticas, desafios enfrentados e possíveis aplicações práticas dessa tecnologia em diferentes contextos da indústria da moda e da fotografia comercial.

No próximo capítulo deste trabalhos são apresentadas as referências, onde serão apresentados os fundamentos do machine learning para classificação de imagem utilizado neste projeto. Dentre os modelos apresentados estão *Support Vector Machines* (SVM), Random Forest Classifier e Redes Neurais Convolucionais (CNNs).

No capítulo 3 os métodos e tecnologias escolhidos para desenvolvimento, desde bibliotecas para machine learning para o *Google Colab* e as métricas para avaliação de desempenho, como acurácia, precisão, *recall, F1-score* e Matriz de Confusão. O capítulo 4 apresenta os resultados obtidos, desenvolvendo uma discussão sobre eles a partir do conjunto de métricas selecionadas para validação dos modelos. Tais resultados servem como fundamento para as conclusões apresentadas no capítulo 5.

2.REVISÃO DA LITERATURA

O avanço tecnológico nas últimas décadas tem sido marcado por uma proliferação exponencial de dados visuais em uma variedade de campos, abrangendo desde a medicina e a biologia até a segurança e o transporte. Essa avalanche de informações visuais representa não apenas um desafio, mas também uma oportunidade sem precedentes para a compreensão e ação em grande escala. Nesse cenário dinâmico, as técnicas de machine learning emergem como um conjunto de ferramentas poderosas para lidar com a complexidade e a heterogeneidade desses dados, capacitando sistemas computacionais a aprender com exemplos passados, reconhecer padrões sutis e fazer previsões informadas sobre o futuro, como afirma Zhang (2020).

Dentro do amplo espectro do machine learning, a classificação de imagens se destaca como uma área de pesquisa e aplicação particularmente significativa. A razão para isso é clara: as imagens são uma forma rica e expressiva de comunicação visual, contendo uma riqueza de informações que vão além do que pode ser capturado por dados puramente textuais. No entanto, a interpretação dessas imagens por máquinas requer algoritmos inteligentes capazes de extrair características relevantes e discernir entre uma variedade de classes ou categorias.

Uma das abordagens mais proeminentes na classificação de imagens é o uso de Redes Neurais Convolucionais (CNNs), que se destacam por sua capacidade de aprender automaticamente padrões hierárquicos em diferentes níveis de abstração, como afirmam Khorrami, Paine e Huang (2017). Inspiradas na organização do córtex visual humano, as CNNs se mostraram altamente eficazes em uma variedade de tarefas de visão computacional, incluindo reconhecimento de objetos, detecção de rostos e segmentação de imagens.

Além das CNNs, uma variedade de outros algoritmos e técnicas de machine learning são aplicados à classificação de imagens, cada um com suas próprias vantagens e limitações. Isso inclui métodos clássicos como Support Vector Machines (SVMs), Árvores de Decisão e *Naive Bayes*, bem como abordagens mais recentes baseadas em técnicas de aprendizado profundo, como Redes Neurais Recorrentes (RNNs) e Redes Generativas Adversariais (GANs).

Um dos principais desafios enfrentados na classificação de imagens é a necessidade de grandes conjuntos de dados rotulados para treinar modelos com desempenho satisfatório. Estratégias como aumento de dados, *transfer learning* e pré-processamento de imagens são frequentemente empregadas para lidar com essa questão, permitindo que modelos sejam treinados de forma mais eficiente e com conjuntos de dados menores.

Além dos desafios técnicos, questões éticas e de privacidade também surgem no contexto da classificação de imagens. Isso é especialmente relevante em áreas como reconhecimento facial, onde o uso indiscriminado da tecnologia pode levantar preocupações sobre vigilância em massa, privacidade individual e discriminação algorítmica, como confirmam Magalhães e Guimarães (2023).

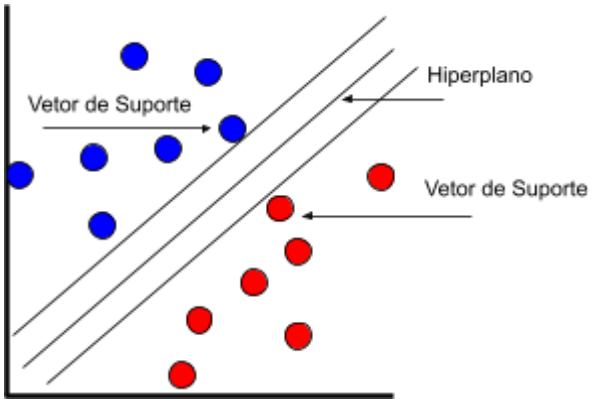
Apesar dos desafios e preocupações, o potencial da classificação de imagens impulsionada por machine learning é imenso. Essa tecnologia tem uma ampla gama de aplicações práticas, incluindo diagnóstico médico, análise de imagens de satélite, detecção de objetos em veículos autônomos, segurança de sistemas de vigilância e outras, todas apresentadas pela redação FIA (FIA BUSINESS SCHOOL, 2021).. À medida que continuamos a avançar no desenvolvimento de algoritmos e técnicas de machine learning, é esperado que o impacto positivo dessas tecnologias em nossa sociedade e economia cresça ainda mais, transformando fundamentalmente a maneira como interagimos com o mundo visual ao nosso redor.

2.1 Support Vector Machines (SVM) para Classificação de Imagens

Support Vector Machines (SVMs) são uma classe de algoritmos de aprendizado supervisionado que se destacam pela sua capacidade de encontrar o hiperplano de separação ótimo em espaços de características de alta dimensionalidade (FACELI et al., 2011). Esses algoritmos têm sido amplamente utilizados em uma variedade de aplicações, desde classificação de texto até reconhecimento de padrões em imagens.

O conceito fundamental por trás dos SVMs é encontrar o hiperplano que maximiza a margem entre as classes, tornando-os especialmente adequados para problemas de classificação binária, como demonstrado na figura 1. Matematicamente, o objetivo do SVM é encontrar o hiperplano $w \cdot x + b = 0$ que separa as classes de forma ótima, onde w é o vetor de pesos e b é o termo de polarização.

Figura 1 - Exemplo de classificador linear, hiperplanos e vetores de suporte



Fonte: Autoria Própria.

A função de decisão para classificar um novo exemplo x é dada por $f(x) = w \cdot x + b$. Se $f(x) > 0$, o exemplo é atribuído à classe positiva, e se $f(x) > 0$, é atribuído à classe negativa.

A margem do hiperplano é dada por $\frac{2}{||w||}$, e o objetivo da otimização é minimizá-la sujeita à restrição de que todos os exemplos de treinamento estejam do lado correto do hiperplano, ou seja, $y_i(w \cdot x_i + b) \geq 1$. O parâmetro de regularização C controla a penalidade por erros de classificação.

Além da classificação linear, os SVMs podem ser estendidos para problemas não lineares através do uso de funções de kernel (BURGES, 1998)., como o kernel polinomial ou o kernel de função de base radial (RBF). Esses kernels transformam os dados para um espaço de características de maior dimensionalidade, onde as classes podem ser separadas de forma mais eficaz.

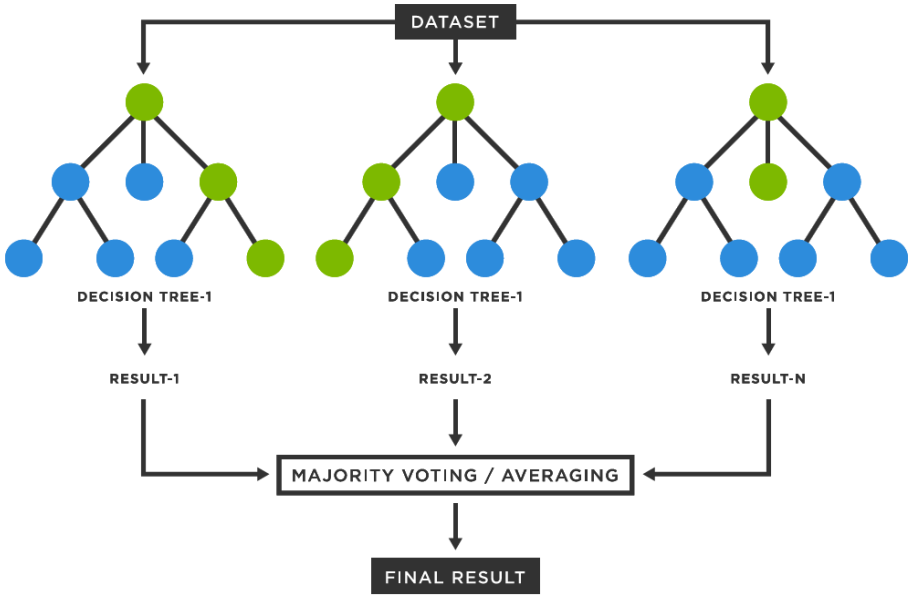
No contexto da classificação de imagens, os SVMs têm sido amplamente utilizados em tarefas como reconhecimento facial, detecção de objetos e segmentação de imagens, como afirmam Andreola e Haertel (2010). Para isso, técnicas como Histogramas de Gradientes Orientados (HOG) e Descritores de Recursos Locais Binários (LBP) são comumente empregadas para extrair características discriminativas das imagens.

Apesar da eficácia demonstrada dos SVMs em várias aplicações de classificação de imagens, eles enfrentam desafios em lidar com conjuntos de dados muito grandes devido à complexidade computacional de encontrar o hiperplano de separação ótimo. Além disso, a seleção cuidadosa de parâmetros, como C e o tipo de kernel, é crucial para o desempenho do modelo.

2.2 Random Forest para Classificação de Imagens

O algoritmo Random Forest Classifier é uma técnica sofisticada de aprendizado de máquina que pertence à classe de modelos baseados em *ensemble*. Ele se destaca por sua capacidade de combinar múltiplas árvores de decisão para realizar a classificação de dado. Cada árvore é construída a partir de uma amostra aleatória do conjunto de treinamento e utiliza uma seleção aleatória de características em cada nó de decisão (KHAN et al., 2021), como demonstrado na figura 2. Durante a fase de inferência, as previsões de cada árvore são agregadas por votação para determinar a classe final de um objeto.

Figura 2 - exemplo gráfico de Random Forest Classifier



Fonte: Gunay (2023).

A história do Random Forest remonta à década de 2000, quando foi proposto por Leo Breiman e Adele Cutler como uma extensão do algoritmo de árvores de decisão, (BREIMAN, 2001). Desde então, tornou-se uma das técnicas de classificação mais populares devido à sua robustez e capacidade de lidar com uma variedade de problemas de aprendizado de máquina.

Uma das principais vantagens do Random Forest Classifier é sua capacidade de lidar efetivamente com conjuntos de dados grandes e de alta dimensionalidade. Ele também é resistente a valores discrepantes e ruído nos dados, graças à agregação de múltiplas árvores.

Para aplicar o Random Forest à classificação de imagens, as características das imagens geralmente são extraídas primeiro. Isso pode incluir descritores de textura, histogramas de cores, características de borda ou características extraídas de redes

neurais convolucionais pré-treinadas. Essas características são então utilizadas como entrada para o modelo de Random Forest, que aprende a mapear as características para as classes de saída.

O treinamento do Random Forest envolve a construção de múltiplas árvores de decisão a partir de amostras aleatórias do conjunto de dados de treinamento. Durante a construção de cada árvore, a seleção aleatória de características em cada nó de decisão ajuda a reduzir a correlação entre as árvores individuais e promove a diversidade do conjunto.

Durante a fase de inferência, as previsões de cada árvore são combinadas por votação para determinar a classe final de uma imagem. O modelo final do Random Forest é capaz de capturar relações complexas e não lineares entre as características das imagens, tornando-o adequado para uma variedade de tarefas de classificação de imagens.

Em resumo, o Random Forest Classifier é uma ferramenta versátil e robusta para a classificação de imagens, oferecendo uma combinação única de capacidade de lidar com grandes conjuntos de dados, resistência a ruídos e capacidade de capturar relações complexas entre características de imagem. Sua aplicabilidade se estende a uma variedade de campos, incluindo reconhecimento de padrões, diagnóstico médico e análise de dados. Com o avanço contínuo da tecnologia, espera-se que o Random Forest continue sendo uma ferramenta valiosa para a análise de dados e tomada de decisões em uma ampla gama de aplicações.

A eficácia do Random Forest em lidar com conjuntos de dados complexos e variados tem sido amplamente demonstrada em diversas áreas. Por exemplo, na área de saúde, ele é usado para prever doenças com base em dados médicos, como histórico de pacientes e resultados de exames (SILVA; SILVA NETO, 2022). Em finanças, o Random Forest é empregado para detecção de fraudes em transações bancárias e previsão de tendências de mercado (SZYSZKA, 2018). Além disso, em aplicações de segurança, como sistemas de vigilância e reconhecimento facial, o Random Forest desempenha um papel importante na identificação e classificação de objetos e pessoas (BOSCH; ZISSERMAN; MUÑOZ, 2007).

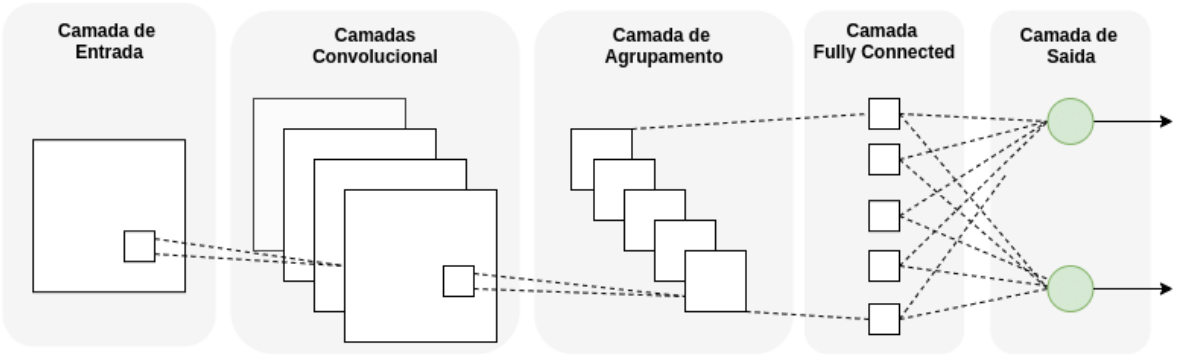
No entanto, apesar de suas vantagens, o Random Forest não é uma solução universal e pode não ser adequado para todos os tipos de problemas de classificação. Por exemplo, em conjuntos de dados muito desequilibrados, onde uma classe é significativamente mais prevalente do que outras, o desempenho do Random Forest pode ser prejudicado (BREIMAN, 2001). Além disso, sua eficácia pode ser limitada em problemas com características altamente correlacionadas ou quando os dados são esparsos.

Em conclusão, o Random Forest Classifier é uma ferramenta poderosa e versátil para a classificação de dados em uma variedade de domínios. Sua capacidade de lidar com grandes conjuntos de dados, resistência a ruídos e capacidade de capturar relações complexas o tornam uma escolha popular para uma ampla gama de aplicações. No entanto, é importante considerar cuidadosamente o contexto específico de cada problema ao decidir se o *Random Forest* é a melhor abordagem, levando em consideração suas vantagens, limitações e requisitos de implementação.

2.3 Redes Neurais Convolucionais (CNNs) para Classificação de Imagens

As Redes Neurais Convolucionais surgiram como uma das técnicas mais revolucionárias e poderosas para a classificação de imagens nas últimas décadas. Inspiradas pela organização do córtex visual biológico, as CNNs conseguem aprender representações hierárquicas de características visuais diretamente dos dados brutos, eliminando a necessidade de uma extração manual de características. Como demonstrado na figura 3.

Figura 3 - exemplo gráfico de Rede Neural Convolucional



Fonte: Barbosa et al. (2021)

Uma CNN é composta por várias camadas, cada uma desempenhando um papel específico no processo de aprendizado e classificação. As camadas convolucionais são responsáveis por aplicar filtros convolucionais para extrair características locais das imagens (GOODFELLOW; BENGIO; COURVILLE, 2016). Matematicamente, a operação de convolução pode ser expressa como uma soma ponderada dos valores dos *pixels* da imagem de entrada, ponderada pelos valores do filtro conforme a Equação 1:

$$S(i,j) = (I.K)(i,j) \; = \; \Sigma_m \Sigma_n I(m,n).K(i - m, j - n) \quad (1)$$

onde $S(i,j)$ é o valor do *pixel* na posição (i,j) da imagem resultante, I é a imagem de entrada e K é o filtro convolucional.

As camadas de *pooling* têm a função de reduzir a dimensionalidade das características e aumentar a invariância a pequenas variações de posição. A equação 2 apresenta o *max-pooling* é uma operação comum nesse tipo de camada, onde é selecionado o valor máximo em uma região de pooling, ajudando a preservar as características mais importantes da imagem:

$$O(i,j) = max_{m,n} I(si + m, sj + n) \quad (2)$$

onde $O(i,j)$ é o valor do *pixel* na posição (i,j) da imagem de saída, I é a imagem de entrada e si e sj são as coordenadas do canto superior esquerdo da região de pooling.

Já as camadas totalmente conectadas são responsáveis por combinar as características aprendidas para realizar a classificação final. Cada neurônio em uma camada totalmente conectada recebe entradas de todos os neurônios na camada anterior e calcula uma saída usando uma função de ativação, como a função ReLU, que introduz não linearidades na rede.

Embora a história das CNNs remonte aos anos 1980, foi apenas na década de 2010 que elas se tornaram amplamente adotadas, impulsionadas pelo aumento da disponibilidade de conjuntos de dados de imagens rotuladas e pelo avanço computacional. Desde então, as CNNs têm sido aplicadas com sucesso em uma variedade de tarefas de visão computacional, incluindo classificação de imagens, detecção de objetos e segmentação semântica.

Um dos principais desafios das CNNs é o treinamento em conjuntos de dados grandes e complexos, que requerem uma quantidade significativa de dados de treinamento e poder computacional. Além disso, o ajuste fino dos hiperparâmetros, como a arquitetura da rede e a taxa de aprendizado, é crucial para o desempenho do modelo.

Durante o treinamento, os pesos da rede são ajustados iterativamente usando algoritmos de otimização, como o gradiente descendente estocástico (SGD), para minimizar uma função de perda. Este é o algoritmo chamado *Backpropagation*.

Inicialmente a imagem de entrada é passada através da rede, camada por camada, até gerar uma predição final. Em cada camada, as operações de convolução,

pooling e ativação são aplicadas para transformar a imagem de entrada em um conjunto de características de alto nível.

A função de perda mais comum para problemas de classificação é a entropia cruzada categórica (Equação 3), que mede a diferença entre a distribuição de probabilidade prevista pela rede e a distribuição real dos rótulos:

$$L\left(y, \widehat{y}\right) = -\ \Sigma_i y_i log(\widehat{y}_i) \tag{3}$$

onde y é o vetor de rótulo verdadeiro (*one-hot encoded*) e $\widehat{y}$ é o vetor de saída previsto pela rede.

Redes Neurais Convolucionais representam uma abordagem poderosa e eficaz para a classificação de imagens, oferecendo a capacidade de aprender representações de características diretamente dos dados brutos e alcançar desempenho de classe mundial em uma variedade de conjuntos de dados de imagens. Seu impacto tem sido significativo em campos como reconhecimento de objetos, diagnóstico médico assistido por computador e análise de dados visuais.

3.MATERIAIS E MÉTODOS

3.1.Keras/Tensorflow

Keras é uma biblioteca de código aberto de alto nível para redes neurais, escrita em Python, que é capaz de rodar sobre o TensorFlow (GÉRON, 2017), CNTK ou Theano. Ela foi desenvolvida para facilitar a experimentação rápida e fácil com redes neurais profundas, sendo especialmente adequada para iniciantes e pesquisadores que desejam construir e treinar modelos de aprendizado profundo de forma eficiente.

TensorFlow, por sua vez, é uma biblioteca de *software* de código aberto para computação numérica e simbólica, desenvolvida pelo Google Brain Team. Ele fornece uma infraestrutura flexível e extensível para tarefas de aprendizado de máquina e *deep learning*, permitindo a criação e treinamento de modelos complexos de forma eficiente, em GPUs e CPUs, para uma ampla gama de aplicativos.

No código, o Keras é utilizado em conjunto com o TensorFlow para construir e treinar modelos de CNNs. A biblioteca Keras é importada para criar modelos de redes neurais camada por camada de forma simples e intuitiva, enquanto o TensorFlow fornece a base de execução eficiente para realizar cálculos numéricos e treinamento de modelos.

Na implementação específica do código, são definidos modelos de CNNs utilizando a interface do Keras, onde são adicionadas camadas de convolução, *pooling, flatten* e *dense* para construir arquiteturas de rede neural. Em seguida, os modelos são compilados e treinados com os dados de entrada, utilizando o TensorFlow como *backend* para executar os cálculos de treinamento e atualização dos pesos da rede.

Portanto, a combinação de Keras e TensorFlow neste código permite a construção, treinamento e avaliação de modelos de aprendizado profundo de forma eficiente e intuitiva (GÉRON, 2017), facilitando a experimentação com diferentes arquiteturas de rede e algoritmos de treinamento para a tarefa específica de classificação de imagens de moda.

3.2. Scikit-learn (Sklearn)

Scikit-learn, comumente referido como sklearn (PEDREGOSA, 2011), é uma biblioteca de aprendizado de máquina de código aberto para Python. Ela oferece uma vasta gama de algoritmos para tarefas de aprendizado supervisionado e não supervisionado, além de ferramentas para pré-processamento de dados, seleção de características e avaliação de modelos.

Em resumo, o scikit-learn desempenha um papel fundamental em diversas etapas do ciclo de vida de um modelo de aprendizado de máquina, desde o pré-processamento dos dados até a avaliação do desempenho do modelo construído.

No código, o sklearn é utilizado em várias etapas do processo de construção e avaliação de modelos de aprendizado de máquina. Ele é empregado para converter rótulos de classes em números inteiros ou criar variáveis *dummy* para variáveis categóricas. Além disso, oferece implementações eficientes de algoritmos como Support Vector Machine e Random Forest para treinamento de modelos.

A biblioteca também fornece métricas de avaliação de desempenho, como precisão, acurácia, *recall, F1-score* e matriz de confusão, que são usadas para avaliar a qualidade dos modelos construídos. Além disso, é utilizada para realizar validação cruzada com a classe *KFold*, que permite dividir os dados em *folds* para avaliar a capacidade de generalização do modelo.

3.3. K-fold Cross Validation

K-fold Cross Validation é uma técnica comum de validação de modelo usada em aprendizado de máquina para avaliar a capacidade de generalização de um modelo em diferentes conjuntos de dados, como definido por Anthony e Holden (1998).

Nesta técnica, o conjunto de dados é dividido em k subconjuntos (ou "*folds*") de tamanho aproximadamente igual. O modelo é treinado k vezes, usando k-1 subconjuntos como dados de treinamento em cada iteração e o subconjunto restante como dados de teste. Esse processo é repetido k vezes, de modo que cada subconjunto seja usado como dados de teste exatamente uma vez.

Ao final, são calculadas métricas de desempenho (como precisão, acurácia, *recall, F1-score*) para cada iteração e, em seguida, calculada a média dessas métricas para obter uma estimativa mais confiável do desempenho do modelo.

No código fornecido, o *K-fold Cross Validation* é utilizado para treinar e avaliar os modelos construídos. A função *train_and_evaluate* implementa essa técnica,

dividindo os dados em *folds*, treinando o modelo em cada *fold* e avaliando seu desempenho usando métricas como precisão, *recall* e *F1-score*. Essa abordagem permite uma avaliação mais robusta do modelo, pois ele é testado em múltiplos conjuntos de dados distintos.

3.4. Métricas de Avaliação de Desempenho

3.4.1. Acurácia

A acurácia é uma métrica comum de desempenho utilizada para avaliar a precisão de um modelo de classificação. Ela representa a proporção de previsões corretas feitas pelo modelo em relação ao número total de previsões. Matematicamente, a acurácia é definida pela equação 4 abaixo.

$$Acurácia = \frac{Número\ de\ Previsões\ Corretas}{Número\ Total\ de\ Previsões} \quad (4)$$

3.4.2. Precisão

A precisão é uma métrica de avaliação comumente usada em problemas de classificação para medir a proporção de instâncias classificadas corretamente como positivas em relação ao total de instâncias classificadas como positivas. Matematicamente, a precisão é definida pela equação 5 abaixo.

$$Precisão = \frac{Verdadeiros\ Positivos}{Verdadeiros\ Positivos + Falsos\ Positivos} \quad (5)$$

3.4.3. Recall

O *recall*, também conhecido como taxa de verdadeiros positivos ou sensibilidade, é uma métrica de avaliação comumente usada em problemas de classificação. Ele mede a proporção de instâncias positivas que foram corretamente identificadas pelo modelo em relação ao total de instâncias positivas no conjunto de dados. Matematicamente, o *recall* é definido pela equação 6 abaixo.

$$Recall = \frac{Verdadeiros\ Positivos}{Verdadeiros\ Positivos + Falsos\ Negativos} \quad (6)$$

3.4.4.F1-Score

O *F1-score* é uma métrica de avaliação que combina precisão e recall em um único número, fornecendo uma medida mais equilibrada do desempenho de um modelo de classificação. É especialmente útil quando há um desequilíbrio entre as classes no conjunto de dados. Matematicamente, o *F1-score* é calculado como a média harmônica da precisão e do *recall*, demonstrado na Equação 7.

$$F1 - score = 2 \times \frac{Precisão \times Recall}{Precisão + Recall} \quad (7)$$

O *F1-score* varia de 0 a 1, onde um valor mais próximo de 1 indica um modelo com melhor desempenho na classificação das instâncias positivas. Ele atinge o seu valor máximo quando a precisão e o *recall* têm valores iguais.

3.4.5.Matriz de Confusão

A matriz de confusão é uma tabela que permite visualizar o desempenho de um modelo de classificação em um conjunto de dados, comparando as classes reais com as classes previstas pelo modelo. Ela mostra o número de verdadeiros positivos (VP), falsos positivos (FP), verdadeiros negativos (VN) e falsos negativos (FN), segundo Faceli *et al.* (2011).. A matriz de confusão tem a seguinte estrutura:

Figura 4 - Matriz de confusão.

	Classe Positiva	Classe Negativa
Classe Positiva	VP	FN
Classe Negativa	FP	VN

Fonte: Autoria própria.

No código anterior, a matriz de confusão foi utilizada para avaliar o desempenho dos modelos de classificação após cada *fold* da validação cruzada. Após prever as classes das amostras de teste, a matriz de confusão foi calculada usando a função *confusion_matrix* da biblioteca Scikit-learn. Em seguida, a matriz de confusão foi exibida graficamente usando a biblioteca Matplotlib para visualizar como o modelo classificou corretamente e incorretamente as amostras de teste. Isso permitiu uma análise visual do desempenho do modelo em cada *fold* da validação cruzada.

4.RESULTADOS

Os experimentos foram conduzidos pelo Google Colab, selecionado devido à sua excepcional facilidade de acesso a recursos computacionais de alto desempenho, incluindo unidades de processamento gráfico (GPUs) e unidades de processamento tensorial (TPUs). Esses recursos são essenciais para o treinamento eficiente de modelos de aprendizado profundo, como Redes Neurais Convolucionais, que são conhecidas por sua intensidade computacional.

A escolha do ambiente de computação em nuvem proporcionou uma série de benefícios significativos. Em primeiro lugar, eliminou a necessidade de investimento em *hardware* especializado, que pode ser caro e exigir manutenção contínua. Em vez disso, os recursos computacionais foram disponibilizados sob demanda, permitindo escalabilidade flexível de acordo com os requisitos do projeto.

Além disso, o Google Colab oferece integração perfeita com outras ferramentas e bibliotecas de *software* amplamente utilizadas em projetos de aprendizado de máquina e ciência de dados. Isso inclui o TensorFlow e o PyTorch, dois dos *frameworks* de *deep learning* mais populares, além de uma variedade de outras bibliotecas para visualização de dados, pré-processamento e avaliação de modelos.

Além disso, o ambiente do *Colab* oferece suporte a bibliotecas de machine learning e ciência de dados, como Scikit-learn, Keras e Pandas, que são essenciais para a implementação e análise de modelos de aprendizado de máquina.

4.1.Dataset

O *dataset* utilizado neste estudo é o resultado de uma seleção e coleta de fotografias, realizadas pelo próprio autor em diversas ocasiões com direitos reservados devido aos direitos compartilhados com as empresas que contrataram os serviços fotográficos. As imagens foram obtidas durante ensaios fotográficos, nos quais o autor desempenhou o papel de fotógrafo, garantindo assim a autenticidade e a qualidade dos dados. Essas fotografias capturam uma variedade de poses, abrangendo uma ampla gama de ângulos, compondo 550 fotos, sendo 198 da classe “casual” e 352 da classe “country”. Todas as imagens foram exportadas no perfil de cores sRGB com dimensões 1920 *pixels* na aresta maior por 1280 *pixels* no formato *png* em 8 *bits*.

Todas as fotografias foram tiradas utilizando o mesmo equipamento fotográfico - Canon SL3 com lente Sigma 18-35mm 1.8 acoplada. A modelo posada em fundo branco sendo iluminada com flashes tochas.

Cada fotografia do dataset foi analisada e rotulada manualmente pelo autor, atribuindo a elas uma das duas classes de interesse para a tarefa de classificação: casual (figura 5.a) ou *country* (figura 5.b). Esses rótulos foram atribuídos com base no estilo do figurino e contexto das imagens, refletindo a distinção fundamental entre os dois tipos de fotografia.

Figura 5 - Fotografias de modelos que representam a classe casual (a) e a classe *country* (b), respectivamente

(a) (b)

Fonte: Autoria própria.

As fotografias rotuladas como "casual" retratam roupas informais, utilizadas no dia a dia de mulheres localizadas no interior paulista. Por outro lado, as fotografias rotuladas como "*country*" apresentam estilos comumente conhecidos pela cultura sertaneja, com franjas, chapéus e outros elementos culturalmente associados ao estilo.

É importante ressaltar que a seleção e rotulagem das fotografias foram realizadas com base na experiência e no conhecimento do autor sobre os estilos e temas abordados. Cada imagem foi cuidadosamente analisada para garantir a precisão e consistência dos rótulos atribuídos, assegurando assim a qualidade e confiabilidade do dataset para fins de análise e experimentação.

4.2. Setup dos Experimentos

Para avaliar o desempenho dos modelos de classificação de imagens, empregamos a técnica de validação cruzada *k-fold*. Foram realizados experimentos com três valores diferentes de k = 3, 5 e 10, para k = 2 e 7, os resultados foram apresentados no apêndice abaixo pois não oferecem variabilidade que corroborem para desenvolver a discussão. Isso significa que o conjunto de dados foi dividido em k partes iguais, e o modelo foi treinado e avaliado k vezes, cada vez usando uma parte diferente como conjunto de teste e as demais como conjunto de treinamento.

Os experimentos foram conduzidos exclusivamente no ambiente de computação em nuvem do Google Colab. Essa escolha foi motivada pela necessidade de utilizar recursos computacionais de alto desempenho, como GPUs, para treinar os modelos de aprendizado profundo de maneira eficiente. O Google Colab oferece acesso gratuito a GPUs Tesla K80 e T4, que são amplamente utilizadas para acelerar o treinamento de modelos de aprendizado profundo.

As imagens foram armazenadas na conta universitária do Google Drive do próprio autor, gerando os diretórios para o acesso dentro do Colab.

4.3. Resultados Obtidos

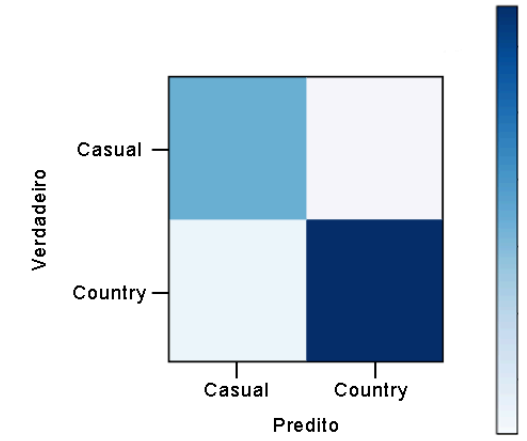
Foram testados diferentes modelos de classificação de imagens, incluindo SVM com *kernel* linear, Random Forest Classifier com 100 estimadores e três arquiteturas diferentes de CNNs devido aos resultados se apresentarem mais consistentes.

4.3.1. SVM

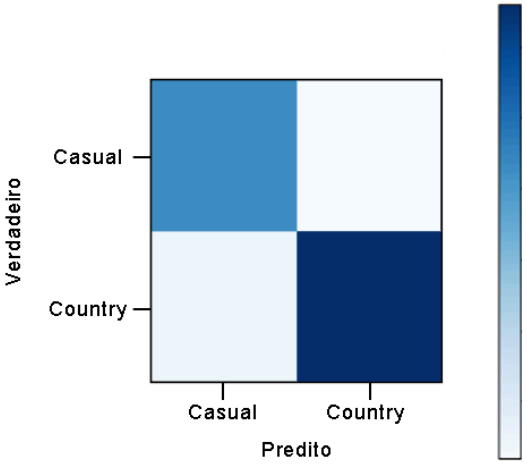
Este modelo de SVM foi avaliado em termos de sua capacidade de separar linearmente as duas classes de imagens. Os resultados de acurácia, precisão, *recall* e *F1-score* foram registrados para análise.
Para avaliar o desempenho do SVM, utilizamos a técnica de *K-fold cross-validation* com três diferentes valores de k iguais a 3, 5 e 10. Essa abordagem nos permite avaliar a robustez do modelo e a variabilidade dos resultados obtidos. A tabela 1 apresenta os valores médios das métricas, assim como seus desvios padrões (DP).

A Figura 6 apresenta as matrizes de confusão para cada valor de k utilizados na validação do SVM.

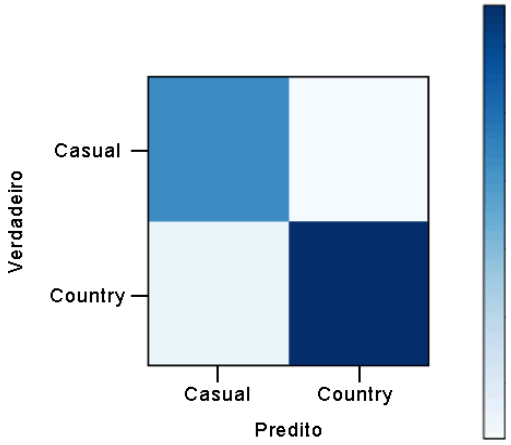
Figura 6 - Matriz de confusão do modelo SVM para k = 3 (a), 5 (b) e 10 (c)



(a)



(b)



(c)

Fonte: Autoria Própria.

Tabela 1 - Valores médios das métricas resultantes do modelo SVM

	k = 3		k = 5		k = 10	
Métrica	Média	DP	Média	DP	Média	DP
Acurácia	0,8852	0,0132	0,9364	0,0244	0,9273	0,0219
Precisão	0,8849	0,0147	0,9369	0,0234	0,9284	0,0218
Recall	0,8852	0,0132	0,9364	0,0244	0,9273	0,0219
F1-Score	0,8851	0,0139	0,9365	0,0245	0,9269	0,0219

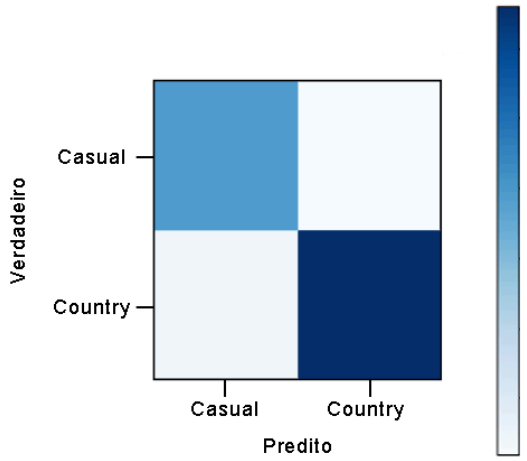
Fonte: Autoria própria.

4.3.2.Random Forest Classifier

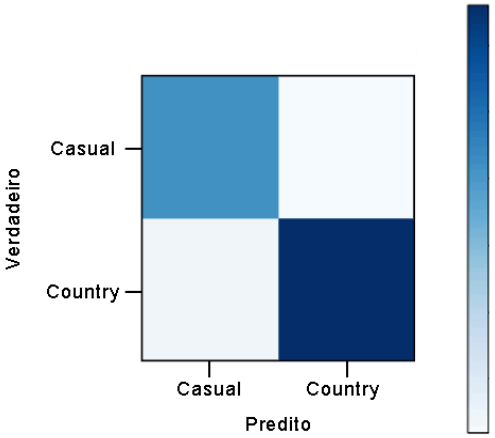
O algoritmo Random Forest Classifier foi treinado com 100 estimadores para construir um conjunto de árvores de decisão aleatórias e avaliado em seu desempenho de classificação de imagens. A tabela 2 apresenta os valores médios das métricas, assim como seus desvios padrões (DP).

Para a classificação por Random Forest Classifier, os gráficos foram apresentados na Figura 7 para cada valor de k abaixo.

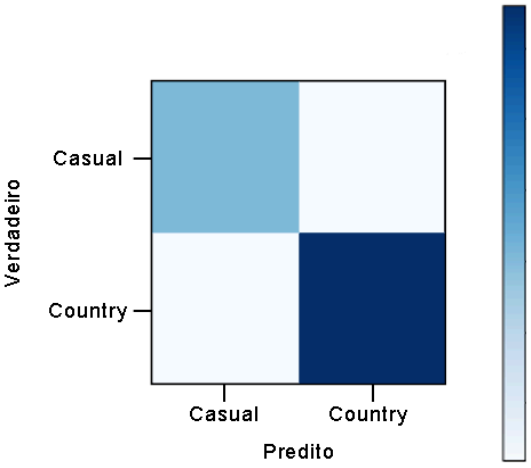
Figura 7 - Matriz de confusão do modelo Random Forest Classifier para k = 3 (a), 5 (b) e 10 (c)



(a)



(b)



(c)

Fonte: Autoria própria.

Tabela 2 - Valores médios das métricas resultantes do modelo de Random Forest Classifier

	k = 3		k = 5		k = 10	
Métrica	Média	DP	Média	DP	Média	DP
Acurácia	0,9728	0,0068	0,9727	0,025	0,9818	0,0234
Precisão	0,9731	0,0064	0,9732	0,0254	0,9825	0,0234
Recall	0,9728	0,0068	0,9727	0,025	0,9818	0,0234
F1-Score	0,9727	0,0064	0,9728	0,0252	0,9818	0,0234

Fonte: Autoria própria.

4.3.3.CNNs

Três arquiteturas diferentes de redes neurais convolucionais foram testadas. Cada uma dessas CNNs tinha uma estrutura ligeiramente diferente, com variações no número de camadas convolucionais, camadas de *pooling* e camadas densas. Os resultados de desempenho foram registrados para cada uma das CNNs e seus valores médios apresentados na tabela 3 abaixo.

Tabela 3 - Tabela com valores médios das métricas de cada Rede Neural Convolucional e seus desvios padrões

	Métrica	**CNN 1**	**DP**	**CNN 2**	**DP**	**CNN 3**	**DP**
K=10	Acurácia	0,902	0,076	0,640	0,032	0,945	0,043
	Precisão	0,910	0,067	0,411	0,041	0,948	0,041
	Recall	0,902	0,076	0,640	0,032	0,945	0,043
	F1-score	0,902	0,075	0,500	0,040	0,944	0,044
K=5	Acurácia	0,640	0,053	0,638	0,030	0,964	0,025
	Precisão	0,412	0,067	0,450	0,081	0,964	0,024
	Recall	0,640	0,053	0,638	0,030	0,964	0,025
	F1-score	0,501	0,066	0,516	0,045	0,964	0,025
K=3	Acurácia	0,922	0,065	0,938	0,019		
	Precisão	0,924	0,062	0,943	0,017		
	Recall	0,922	0,065	0,938	0,019		
	F1-score	0,919	0,069	0,938	0,020		

Fonte: Autoria própria.

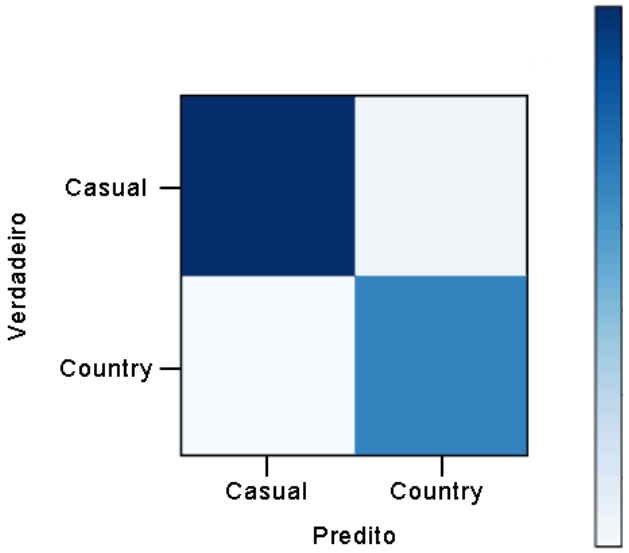
CNN 1: Esta CNN possui duas camadas convolucionais seguidas de camadas de *max-pooling* para redução da dimensionalidade. Após as camadas convolucionais,

há uma camada densa com 64 unidades, seguida por uma camada de saída com ativação *softmax* para classificação.

Abaixo foram apresentadas as matrizes de confusão referentes aos *folds* com os melhores resultados em termos de acurácia para cada valor de k.

Para k=3 os resultados em *fold* 4 realizou as classificações com grande precisão o estilo *country* e identificou de maneira concisa o estilo casual. Conforme a figura 8 abaixo. A acurácia, precisão, *recall* e *F1-score* foram de 99,1%.

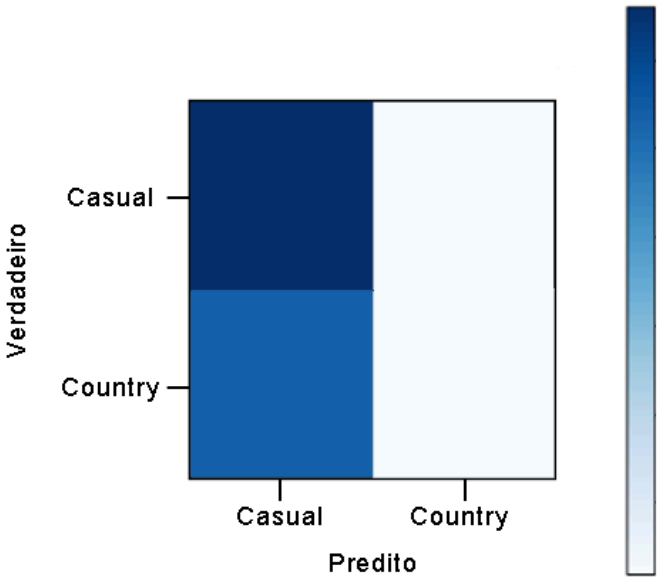
Figura 8 - CNN 1 Matriz de confusão em *fold* 4 para k=3



Fonte: Autoria própria.

Para k=5 os resultados em *fold* 5 foi extremamente preciso em classificar o estilo casual e o estilo *country*. Conforme a figura 9 abaixo. A acurácia foi de 55,45%, precisão de 30,75%, *recall* de 55,45% e *F1-score* igual a 39,56%.

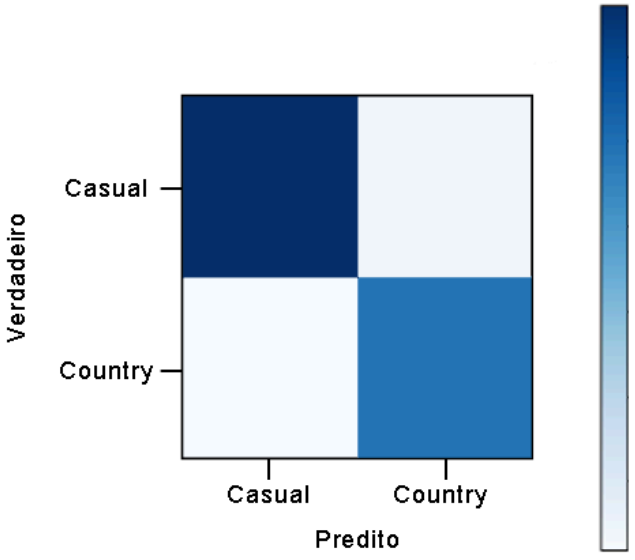
Figura 9 - CNN 1 Matriz de confusão em *fold* 5 para k=5



Fonte: Autoria própria.

Para k=10 os resultados em *fold* 5 foi extremamente capaz em classificar o estilo *country*, porém falhou em classificar o estilo casual. Conforme a figura 10 abaixo. A acurácia, precisão, *recall* e *F1-score* foram de 100%.

Figura 10 - CNN 1 Matriz de confusão em *fold* 5 para k=10

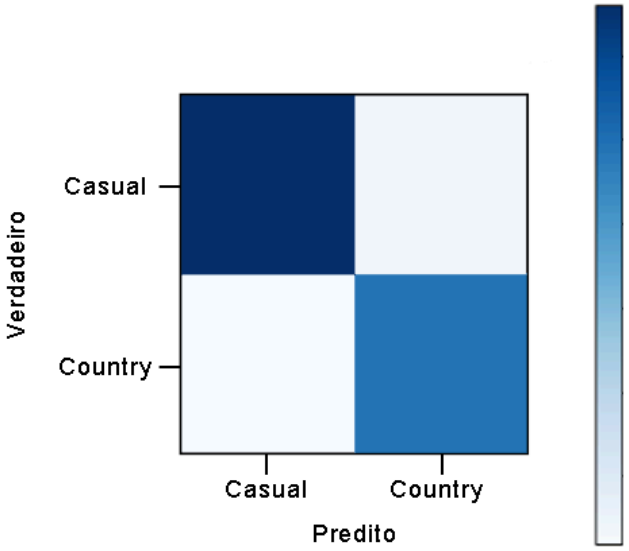


Fonte: Autoria própria.

CNN 2: Esta arquitetura de CNN é mais profunda em comparação com a primeira, apresentando camadas convolucionais e de *max-pooling* adicionais. Após as camadas convolucionais e de *pooling*, há uma sequência de camadas densas, cada vez com menos unidades, culminando em uma camada de saída com ativação *softmax*.

Para k=3, *fold* 5 apresentou os maiores parâmetros, onde foi possível classificar ambos os estilos. Todos os parâmetros correspondem a valores aproximados a 96,4%. Conforme a figura 11 abaixo.

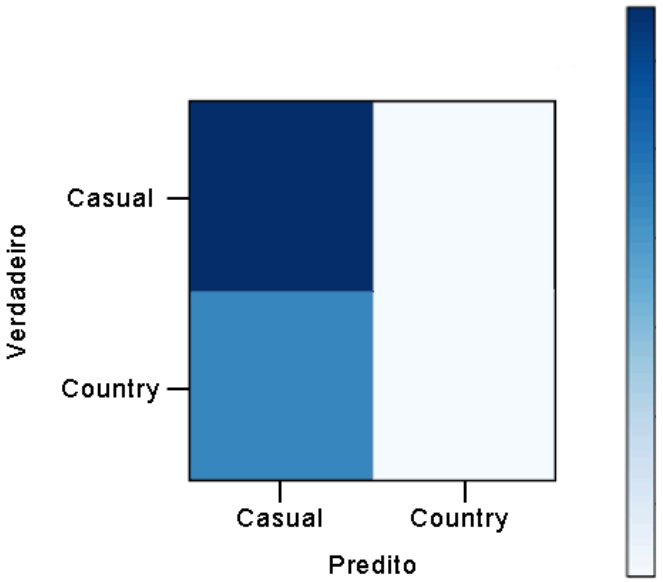
Figura 11 - CNN 2 Matriz de confusão em *fold* 5 para k=3



Fonte: Autoria própria.

Com k=5, resultados semelhantes em *fold* 5 para a CNN 1 se repetiram. Agora com acurácia igual a 60%, precisão igual a 36%, *recall* 60% também e *F1-score* igual a 44,99%. Conforme figura 12 abaixo.

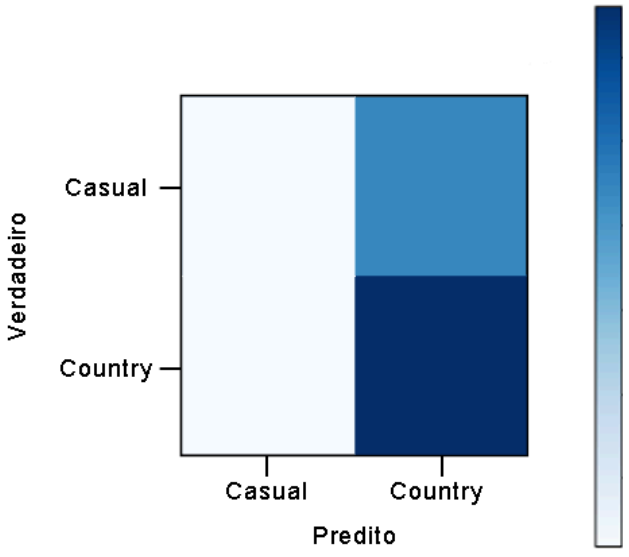
Figura 12 - CNN 2 Matriz de confusão em *fold* 5 para k=5



Fonte: Autoria própria.

Em k=10 foram registrados resultados muito semelhantes de *fold* 4 com *fold* 5 de k=5, onde a classificação foi precisa para o estilo casual, porém possuiu um alto índice de falhas para o estilo *country*. Os valores dos parâmetros foram exatamente iguais aos anteriores, a figura 13 demonstra esta semelhança.

Figura 13 - CNN 2 Matriz de confusão em *fold* 4 para k=10



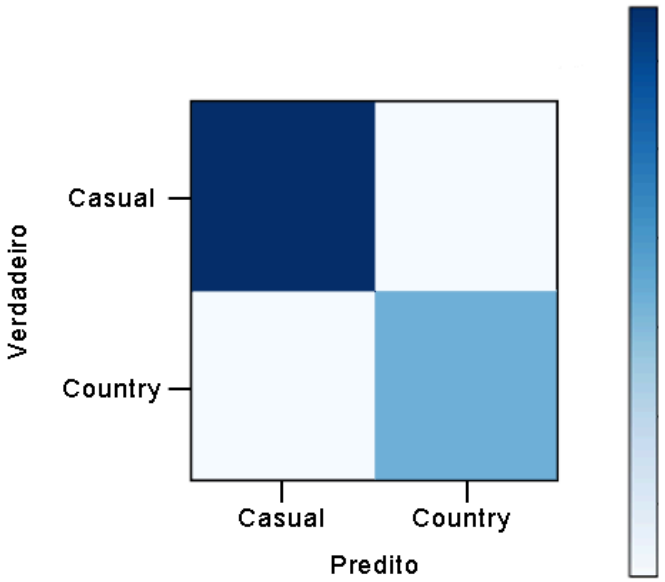
Fonte: Autoria própria.

CNN 3: A terceira CNN é mais complexa, apresentando uma estrutura mais profunda e uma combinação de camadas convolucionais, de *pooling* e uma camada densa

adicional antes da camada de saída. Esta arquitetura visa capturar características mais abstratas e complexas das imagens.

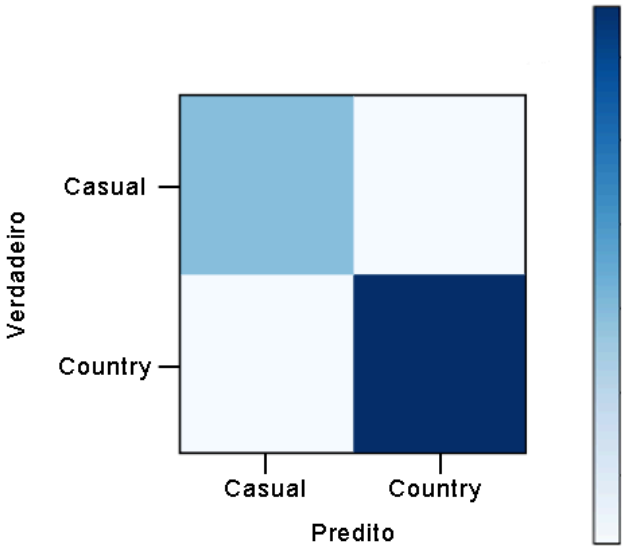
Tanto para k=5 e k=10, nas *folds* 5 e 4 respectivamente foram obtidos os maiores valores dos parâmetros, sendo todos, acurácia, precisão, *recall* e *F1-score*, iguais a 100%. Conforme as figuras 14 e 15 abaixo.

Figura 14 - CNN 3 Matriz de confusão em *fold* 5 para k=5



Fonte: Autoria própria.

Figura 15 - CNN 3 Matriz de confusão em *fold* 5 para k=10



Fonte: Autoria própria.

4.4. Discussão

A coleta e a rotulagem manual do *dataset* pelo autor garantiram a autenticidade e a qualidade dos dados, resultando em uma base confiável para o treinamento dos modelos. A padronização do equipamento utilizado, uma Canon SL3 com lente Sigma 18-35 mm 1.8, e a uniformidade das condições de iluminação ajudaram a minimizar variabilidades indesejadas, assegurando consistência na qualidade das imagens.

A aplicação da técnica de validação cruzada *k-fold* foi essencial para a avaliação robusta dos modelos, proporcionando uma visão abrangente de seu desempenho. A divisão do conjunto de dados em diferentes partes para treinamento e teste permitiu medir a variabilidade dos resultados, aumentando a confiança na generalização dos modelos.

Os resultados obtidos com diferentes modelos de classificação de imagens, incluindo SVM, Random Forest e CNNs, foram variados e ilustraram as capacidades e limitações de cada abordagem.

Para o modelo SVM, os resultados mostraram uma acurácia crescente com o aumento do valor de k, atingindo 93,64% com k=5. Embora o SVM tenha demonstrado um bom desempenho geral, a natureza linear do *kernel* pode ter limitado sua capacidade de capturar a complexidade dos dados visuais. Com diferentes valores de k, o modelo é treinado e testado em diferentes partes dos dados, mas como o *dataset* é bem equilibrado e o modelo é consistente, os resultados médios das métricas tendem a ser semelhantes.

O Random Forest Classifier, por sua vez, apresentou um desempenho também satisfatório, com acurácia chegando a 98,18% para k=10. Esse modelo se beneficiou da combinação de múltiplas árvores de decisão, proporcionando robustez e alta precisão na classificação das imagens. A consistência dos resultados também indica que o *dataset* é bem equilibrado e representativo das classes que estão sendo classificadas. Se houvesse grandes discrepâncias ou desequilíbrios nas classes, os resultados poderiam variar mais entre diferentes *folds*.

As CNNs, com suas diferentes arquiteturas, mostraram resultados variados. A primeira arquitetura de CNN, com duas camadas convolucionais e *max-pooling*, teve um desempenho excepcional em k=3 e k=10. Entretanto, os resultados em k=5 indicam uma queda na precisão para a classe casual. A segunda CNN, mais profunda, apresentou uma melhor generalização, especialmente em k=3, com uma acurácia de 96,4%. Entretanto, para k=5 e k=10, a precisão e o *recall* foram comprometidos, sugerindo que a profundidade adicional não sempre se traduz em melhor desempenho.

A terceira CNN, a mais complexa, foi a que apresentou os melhores resultados em termos de consistência e precisão, alcançando 100% de acurácia, precisão, *recall* e *F1-score* para k=5 e k=10. Isso indica que a estrutura mais profunda e a combinação de várias camadas permitiram capturar características mais abstratas e detalhadas das imagens, resultando em uma classificação mais precisa.

5. CONCLUSÃO

O objetivo deste trabalho é aprimorar o algoritmo de classificação de imagens para identificar classes em tempo real e fornecer referências visuais compatíveis com a produção atual. Isso visa automatizar e reduzir o tempo e esforço dos ensaios fotográficos. A coleta e rotulagem manual garantiram dados autênticos e de qualidade. Destacou-se a terceira CNN, que alcançou 100% de acurácia, precisão, *recall* e *F1-score* para k=5 e k=10, comprovando a eficácia das CNNs na identificação de padrões visuais complexos. Este algoritmo poderia assegurar um funcionamento de um *software* que funcione em tempo real.

Os resultados deste estudo destacam a importância de uma abordagem meticulosa na coleta e preparação dos dados, bem como a escolha cuidadosa de técnicas e modelos para a tarefa de classificação de imagens. A coleta e a rotulagem manual das imagens pelo autor garantiram a autenticidade e a qualidade do dataset, crucial para um treinamento eficaz dos modelos. A padronização do equipamento fotográfico e das condições de iluminação ajudaram a minimizar variações indesejadas, resultando em um *dataset* consistente e de alta qualidade.

Os experimentos realizados neste estudo demonstraram que os modelos de classificação de imagens, incluindo Support Vector Machine, Random Forest Classifier e Redes Neurais Convolucionais, apresentaram desempenho satisfatório na tarefa de distinguir entre fotografias dos tipos casual e *country*. Todos os modelos alcançaram resultados promissores em termos de acurácia, precisão, *recall* e *F1-score*, indicando que são capazes de generalizar bem para novos dados.

Os resultados indicam que a escolha do modelo e a arquitetura apropriada são cruciais para a tarefa de classificação de imagens. Modelos como o Random Forest e a CNN mais complexa (CNN 3) mostraram-se particularmente eficazes. A coleta e preparação meticulosa dos dados, juntamente com a aplicação de técnicas de validação robustas, foram essenciais para alcançar resultados confiáveis e precisos.

As CNNs são eficientes em termos de computação e memória devido ao compartilhamento de pesos, o que facilita o treinamento e a generalização, como visto na segunda CNN, que apresentou uma acurácia de 96,4% em k=3. Elas aprendem automaticamente a extrair características importantes, como bordas e texturas, e são invariantes à translação, reconhecendo características independentemente da posição na imagem. Também preservam a relação espacial entre *pixels*, capturando padrões locais e globais, o que foi crucial para o desempenho das CNNs testadas. Essas vantagens tornam as CNNs extremamente eficazes para a classificação de imagens, demonstrado pelos resultados robustos e precisos obtidos nos experimentos.

Trabalhos Futuros

Existem várias oportunidades para estender este trabalho em pesquisas futuras:

1. Exploração de Arquiteturas de CNN Mais Avançadas: Investigar arquiteturas de CNN mais avançadas, como redes residuais (ResNet), redes neurais convolucionais recorrentes (RCNNs) ou redes neurais adversárias generativas (GANs), para avaliar se elas podem melhorar ainda mais o desempenho na classificação de imagens.

2. Pré-processamento Avançado de Imagens: Explorar técnicas avançadas de pré-processamento de imagens, como aumento de dados, normalização avançada e segmentação de imagens, para melhorar a capacidade dos modelos de aprendizado de máquina em capturar e generalizar padrões visuais complexos.

3. Ajuste de Hiperparâmetros: Realizar uma busca mais abrangente de hiperparâmetros para otimizar o desempenho dos modelos, incluindo a variação de parâmetros como taxa de aprendizado, tamanho do batch e arquitetura da rede.

4. Análise de Interpretabilidade: Investigar técnicas para interpretar e visualizar as decisões dos modelos, como mapas de ativação, saliência de gradiente e caminhos de ativação, a fim de entender melhor quais características visuais são importantes para a classificação de imagens.

5. Aplicação em Outros Domínios: Testar os modelos desenvolvidos em outros conjuntos de dados e domínios de aplicação para avaliar sua capacidade de generalização e adaptabilidade a diferentes problemas de classificação de imagens.

6. Implementação prática: incrementar este algoritmo de forma que o programa identifique a classe em tempo real e retorne referências visuais condizentes com a produção vigente. Desta forma espera-se que o tempo de um ensaio fotográfico, assim como seu esforço, sejam automatizados.

REFERÊNCIAS

ANDREOLA, R.; HAERTEL, V. Classificação de imagens hiperespectrais empregando Support Vector Machines. **Bol. Ciênc. Geod.**, [Curitiba], v. 16, n. 2, p. 210-231, abr./jun. 2010.

ANTHONY, M.; HOLDEN, S. B. Cross-validation for binary classification by realvalued functions: theoretical analysis. In: ANNUAL CONFERENCE ON COMPUTATIONAL LEARNING THEORY, 11., 1998, Madison. **Proceedings** [...] Madison: Association for Computing Machinery, 1998. p. 218-229.

BARBOSA, G. N. N.; BEZERRA, G. M. G.; MEDEIROS, D. S. V.; LOPEZ, M. A.; MATTOS, D. M. F. Segurança em redes 5G: oportunidades e desafios em detecção de anomalias e predição de tráfego baseadas em aprendizado de máquina. *In*: SANTIN, A. O.; ARAUJO, R. S. S.; ABELÉM, A. J. G. (org.). **Minicursos do XXI Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais**. Porto Alegre: Sociedade Brasileira da Computação, 2021. Disponível em: https://books-sol.sbc.org.br/index.php/sbc/catalog/book/71. Acesso em: 21 maio 2024.

BOSCH, A.; ZISSERMAN, A.; MUÑOZ, X. Image Classification using Random Forests and Ferns, In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION, 11., 2007, Rio de Janeiro. **Proceedings** [...]. [Piscataway]: IEEE, 2007. Disponível em: https://ieeexplore.ieee.org/document/4409066. Acesso em: 24 maio 2024.

BREIMAN, L. Random Forests. **Machine Learning**, Netherlands, v. 45, p 5-32, oct. 2001. DOI: https://doi.org/10.1023/A:1010933404324. Disponível em: https://link.springer.com/article/10.1023/A:1010933404324. Acesso em: 24 maio 2024.

BURGES, Christopher JC. A tutorial on support vector machines for pattern recognition. **Data mining and knowledge Discovery**, New York: Springer, v. 2, n. 2, p. 121–167, 1998.

FACELI, K.; LORENA, A. C.; GAMA, J.; CARVALHO, A. C. P. L. F. **Inteligência artificial**: uma abordagem de aprendizado de máquina. Rio de Janeiro: Gen-LTC, 2011.

FIA BUSINESS SCHOOL. **Machine Learning**: como funciona, benefícios, tipos e exemplos. [S. l.]: Fia Business School, 12 nov. 2021. Disponível em: https://fia.com.br/blog/machine-learning/. Acesso em: 28 maio 2024.

GÉRON, A. **Hands-On Machine Learning with Scikit-Learn and TensorFlow**: Concepts, Tools, and Techniques to Build Intelligent Systems. [S.l.]: O'Reilly, 2017.

GOODFELLOW, I; BENGIO, Y; COURVILLE, A. **Deep Learning**. MIT Press. 2016.

GUNAY, Deniz. **Random Forest**. [*S. l.*]: Medium, 11 set. 2023. Disponível em: https://medium.com/@denizgunay/random-forest-af5bde5d7e1e. Acesso em: 20 maio 2024.

KHAN, S.; SANOVAR; KUMAR, S.; KUMAR, H. Credit card fraud detection using machine learning. **International Journal of Scientific and Research Publications**, [New Delhi], v. 11, n. 6, p. 60-67, jun. 2021. Disponível em: https://www.ijsrp.org/research-paper-0621.php?rp=P11411322. Acesso em: 20 maio 2024.

KHORRAMI, P.; PAINE, T. L.; HUANG, T. S. **Do deep neural networks learn facial action units when doing expression recognition?** [S. l.], 16 mar. 2017. Disponível em: https://arxiv.org/abs/1510.02969. Acesso em: 20 maio 2024

NÚMERO de empreendedores da moda cresceu durante crise da covid-19. [S. l.]: Diário do Comércio, 12 nov. 2021. Disponível em: https://dcomercio.com.br/publicacao/s/numero-de-empreendedores-da-moda-cresceu-durante-crise-da-covid-19. Acesso em: 03 mar. 2023.

MAGALHÃES, A. V. A.; GUIMARÃES, F. C. **O reconhecimento facial como instrumento de segurança pública e seus impactos na Lei Geral de Proteção de Dados - LGPD**. 2023. Monografia - Centro Universitário UNIFG, Guanambi, 2023.

PEDREGOSA, Fabian et al. Scikit-learn: Machine Learning in Python. J**ournal Of Machine Learning Research**, v. 12, n. 0, p. 2825-2830, out. 2011.

RIBEIRO, H. L. **Reconhecimento de Gestos Usando Segmentação de Imagens Dinâmicas de Mãos Baseada no Modelo de Mistura de Gaussianas e Cor de Pele**. 2006. Dissertação (Mestrado em Engenharia Elétrica) - Escola de Engenharia de São Carlos da Universidade de São Paulo, São Carlos, 2006.

SILVA, R.; SILVA NETO, D. R. Inteligência artificial e previsão de óbito por Covid-19 no Brasil: uma análise comparativa entre os algoritmos Logistic Regression, Decision Tree e Random Forest. **Saúde em Debate**, Rio de Janeiro, v. 46, n. 8, p. 118-129, dez. 2022. Disponível em: https://www.saudeemdebate.org.br/sed/article/view/7670. Acesso em: 20 maio 2024.

SZYSZKA, P. **RANDOM FOREST**: Uma Investigação da Eficiência de Mercado de Ações da Vale. Dissertação (Mestrado em Economia) - Fundação Getulio Vargas, Escola de Pós-Graduação em Economia. FGV, Rio de Janeiro, 2018.

ZHANG, X. **A Matrix Algebra Approach to Artificial Intelligence**. Singapore: Springer, 2020. p. 223–440.

APÊNDICE A - CÓDIGO FONTE UTILIZADO

```
import os
import numpy as np
import cv2
from sklearn.model_selection import KFold
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, confusion_matrix
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from keras.utils import to_categorical
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from keras.utils import to_categorical

from google.colab.patches import cv2_imshow  # Para exibir imagens no
Colab

# Montar o Google Drive
from google.colab import drive
drive.mount('/content/drive')

num_classes = 2

# Definir os diretórios de onde as imagens serão carregadas
dir1 =
dir2 =
# Adicione quantos diretórios forem necessários

# Definir o número de folds
num_folds = 3

# Função para carregar imagens de um diretório
def load_images(directory, target_size=(128, 128)):
    images = []
    labels = []

    #label_encoder = LabelEncoder()  # Inicializa o codificador de
rótulos
```

```
    #label_encoder.fit(os.listdir(directory))  # Ajusta o codificador
aos nomes dos diretórios

    for filename in os.listdir(directory):
        img = cv2.imread(os.path.join(directory, filename))
        if img is not None:
            img_resized = cv2.resize(img, target_size)
            height, width, channels = img_resized.shape
            img_resized = [x for xs in img_resized for x in xs]
            img_resized = [x for xs in img_resized for x in xs]
            #print("IMG", img_resized)
            images.append(img_resized)
            #print("DIR ", directory)
            #print("DIR -- ", directory.split('/')[-2])
            labels.append(directory.split('/')[-2])  # Supondo que o
nome do diretório seja a classe da imagem

    # Converta os rótulos em números inteiros usando o codificador de
rótulos
    #labels_encoded = label_encoder.transform(labels)

    return images, labels, height, width, channels

# Carregar imagens de todos os diretórios
all_images = []
all_labels = []
for directory in [dir1, dir2]:
    images, labels, height, width, channels = load_images(directory)
    print(labels)
    all_images.extend(images)
    all_labels.extend(labels)

# Converter listas em arrays numpy
all_images = np.array(all_images)
all_labels = np.array(all_labels)

# Definir as métricas
metrics = {
    'accuracy': accuracy_score,
    'precision': precision_score,
    'recall': recall_score,
    'f1_score': f1_score
}
```

```
# Função para treinar e avaliar um modelo com K-fold cross validation
def train_and_evaluate(model, name):

    if "cnn" in name:
      all_images_reshaped = all_images.reshape(-1, height, width,
channels)

    kf = KFold(n_splits=num_folds, shuffle=True)
    results = {metric: [] for metric in metrics}

    fold = 1
    for train_index, test_index in kf.split(all_images):

        #print("TESTE1")
        X_train, X_test = all_images[train_index],
all_images[test_index]
        y_train, y_test = all_labels[train_index],
all_labels[test_index]

        #print("TESTE2")

        # Normalizar as imagens entre 0 e 1
        X_train = X_train.astype('float32') / 255
        X_test = X_test.astype('float32') / 255

        #print("TESTE3")

        # Converter as labels em one-hot encoding
        #y_train = to_categorical(y_train)
        #y_test = to_categorical(y_test)

        #print("TESTE4")

        #print("X ", X_train)
        #print("y ", y_train)

        #X_train = [x for xs in X_train for x in xs]

        #print("X TRAIN", X_train)
        #print("y train", y_train)
        #X_train = np.concatenate(X_train, axis=0)

        # Treinar o modelo
```

```
        model.fit(X_train, y_train)

        # Avaliar o modelo
        y_pred = model.predict(X_test)
        #for metric_name, metric_func in metrics.items():
        #    print("metric", metric_name)
        #    print("y_test", y_pred)

        accuracy = accuracy_score(y_test, y_pred)
        precision = precision_score(y_test, y_pred, average='weighted')
        recall = recall_score(y_test, y_pred, average='weighted')
        f1 = f1_score(y_test, y_pred, average='weighted')

        print("acc", accuracy)
        print("precision", precision)
        print("recall", recall)
        print("f1", f1)

        # Plotar e salvar a matriz de confusão
        cm = confusion_matrix(y_test, y_pred)

        plt.imshow(cm, cmap=plt.cm.Blues)
        plt.title('Matriz de Confusão')
        plt.colorbar()
        plt.xlabel('Predito')
        plt.ylabel('Verdadeiro')
        plt.xticks(range(len(set(all_labels))), set(all_labels))
        plt.yticks(range(len(set(all_labels))), set(all_labels))
        plt.savefig('matriz_de_confusao_' + name + '_' + str(fold) +
'.png')

        fold += 1

    # Calcular média e desvio padrão das métricas
    avg_results = {metric: np.mean(values) for metric, values in
results.items()}
    std_results = {metric: np.std(values) for metric, values in
results.items()}

    # Salvar resultados em um arquivo
    with open('resultados_' + name + '_' + str(num_folds) + '.txt',
'w') as f:
```

```
        f.write('Resultados do ' + name + ' com ' + str(num_folds) +
'-fold cross validation:\n')
        for metric in metrics.keys():
            f.write(metric + ': ' + str(avg_results[metric]) + ' ± ' +
str(std_results[metric]) + '\n')

# Definir e treinar modelos
# Modelo SVM
svm_model = SVC(kernel='linear')
train_and_evaluate(svm_model, 'svm')

# Modelo Random Forest
rf_model = RandomForestClassifier(n_estimators=100)
train_and_evaluate(rf_model, 'random_forest')

# Função para carregar imagens de um diretório
def load_images_cnn(directory, target_size=(128, 128)):
    images = []

    for filename in os.listdir(directory):
        img = cv2.imread(os.path.join(directory, filename))
        if img is not None:
            img_resized = cv2.resize(img, target_size)
            height, width, channels = img_resized.shape
            images.append(img_resized)

    # Converta os rótulos em números inteiros usando o codificador de
rótulos
    #labels_encoded = label_encoder.transform(labels)

    return images

# Carregar imagens de todos os diretórios
all_images = []
for directory in [dir1, dir2]:
    images = load_images_cnn(directory)
    all_images.extend(images)

def train_and_validate_cnn_kfold(model, num_classes, X, y, name,
num_folds=5, epochs=10, batch_size=32):
    kf = KFold(n_splits=num_folds, shuffle=True)
    fold = 1
```

```
    accuracies = []
    precisions = []
    recalls = []
    f1_scores = []

    for train_index, test_index in kf.split(X):
        X_train, X_test = X[train_index], X[test_index]
        y_train, y_test = y[train_index], y[test_index]

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

        # Treinar modelo
        model.fit(X_train, y_train, epochs=epochs,
batch_size=batch_size, verbose=1)

        # Avaliar modelo
        y_pred = model.predict(X_test)
        y_pred_classes = np.argmax(y_pred, axis=1)
        y_test_classes = np.argmax(y_test, axis=1)

        # Calcular métricas
        accuracy = accuracy_score(y_test_classes, y_pred_classes)
        precision = precision_score(y_test_classes, y_pred_classes,
average='weighted')
        recall = recall_score(y_test_classes, y_pred_classes,
average='weighted')
        f1 = f1_score(y_test_classes, y_pred_classes,
average='weighted')

        accuracies.append(accuracy)
        precisions.append(precision)
        recalls.append(recall)
        f1_scores.append(f1)

        # Gerar matriz de confusão
        cm = confusion_matrix(y_test_classes, y_pred_classes)

        # Plotar matriz de confusão
        plt.figure(figsize=(8, 6))
        plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
        plt.title(f'Matriz de Confusão - Fold {fold}')
```

```
        plt.colorbar()
        plt.xlabel('Predito')
        plt.ylabel('Verdadeiro')
        plt.show()

        # Imprimir métricas
        print(f"Fold {fold}:")
        print("Acurácia:", accuracy)
        print("Precisão:", precision)
        print("Recall:", recall)
        print("F1-score:", f1)

        fold += 1

    # Imprimir métricas médias
    print("\nMétricas Médias:")
    print("Acurácia Média:", np.mean(accuracies))
    print("Desvio padrão:", np.std(accuracies))
    print("Precisão Média:", np.mean(precisions))
    print("PDesvio padrão:", np.std(precisions))
    print("Recall Médio:", np.mean(recalls))
    print("Desvio padrão:", np.std(recalls))
    print("F1-score Médio:", np.mean(f1_scores))
    print("Desvio padrão:", np.std(f1_scores))

    # Salvar resultados em um arquivo
    with open('resultados_' + name + '_' + str(num_folds) + '.txt',
'w') as f:
        f.write('Resultados do ' + name + ' com ' + str(num_folds) +
'-fold cross validation:\n')
        f.write("Acurácia Média:" + str(np.mean(accuracies)))
        f.write("Desvio padrão:" + str(np.std(accuracies)))
        f.write("Precisão Média:" + str(np.mean(precisions)))
        f.write("PDesvio padrão:" + str(np.std(precisions)))
        f.write("Recall Médio:" + str(np.mean(recalls)))
        f.write("Desvio padrão:" + str(np.std(recalls)))
        f.write("F1-score Médio:" + str(np.mean(f1_scores)))
        f.write("Desvio padrão:" + str(np.std(f1_scores)))

# Remodelar as imagens para a forma correta
#all_images_reshaped = all_images.reshape(-1, height, width, channels)
all_images = np.array(all_images)
print(all_images.shape)
```

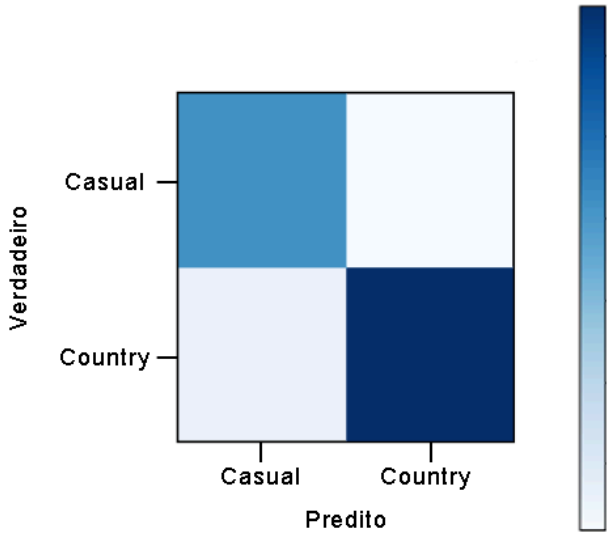
```
from sklearn.preprocessing import LabelBinarizer

from keras.utils import to_categorical

label_map = {label: idx for idx, label in enumerate(set(all_labels))}
all_labels_encoded = [label_map[label] for label in all_labels]
all_labels_encoded = to_categorical(all_labels_encoded)

# Construir modelo CNN 1
model = Sequential([
    Conv2D(32, (3, 3), activation='relu',
input_shape=all_images.shape[1:]),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    Flatten(),
    Dense(64, activation='relu'),
    Dense(num_classes, activation='softmax')
])

train_and_validate_cnn_kfold(model, num_classes, all_images,
all_labels_encoded, "cnn1")

# Construir modelo CNN 2
model = Sequential([
    Conv2D(32, (3, 3), activation='relu',
input_shape=all_images.shape[1:]),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    Flatten(),
    Dense(64, activation='relu'),
    Dense(32, activation='relu'),
    Dense(16, activation='relu'),
    Dense(8, activation='relu'),
    Dense(4, activation='relu'),
    Dense(2, activation='relu'),
    Dense(num_classes, activation='softmax')
])
```

```
train_and_validate_cnn_kfold(model, num_classes, all_images,
all_labels_encoded, "cnn2")

# Construir modelo CNN 2
model = Sequential([
    Conv2D(32, (3, 3), activation='relu',
input_shape=all_images.shape[1:]),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),  # Nova camada de MaxPooling
    Conv2D(128, (3, 3), activation='relu'),  # Nova camada
convolucional
    Flatten(),
    Dense(128, activation='relu'),  # Nova camada densa
    Dense(64, activation='relu'),
    Dense(num_classes, activation='softmax')
])

train_and_validate_cnn_kfold(model, num_classes, all_images,
all_labels_encoded, "cnn3")
```

APÊNDICE B - RESULTADOS PARA K = 2

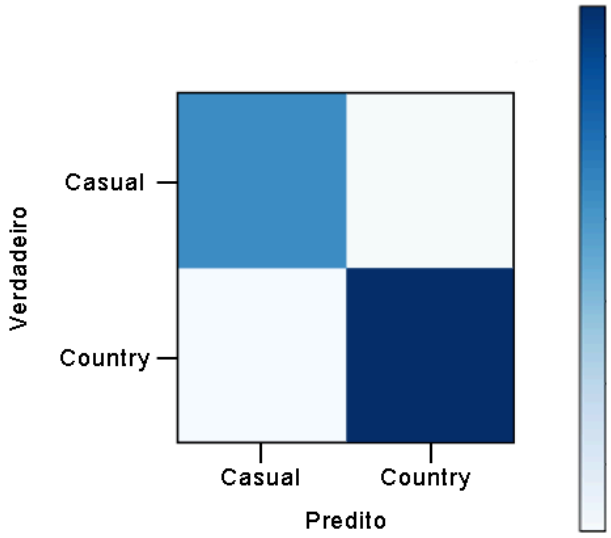
Valores médios das métricas do modelo Random Forest.

K=2		
Métrica	**Média**	**DP**
Acurácia	0,9091	0,0000
Precisão	0,9101	0,0017
Recall	0,9091	0,0000
F1-Score	0,9093	0,0004



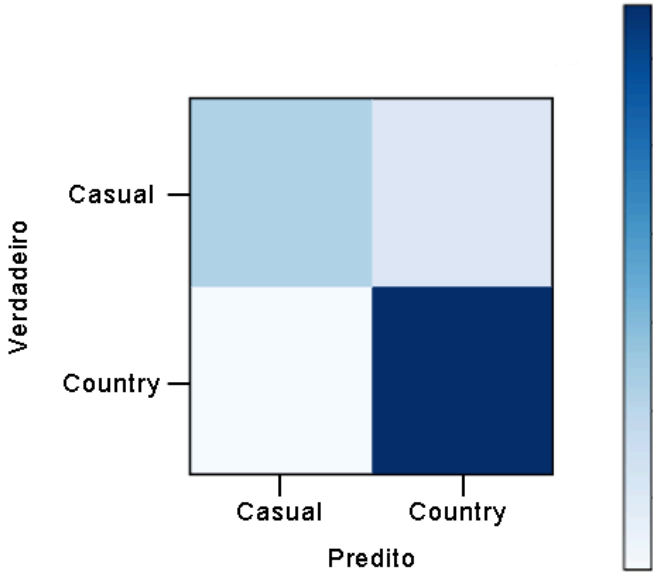
Valores médios das métricas do modelo SVM.

K=2		
Métrica	**Média**	**DP**
Acurácia	0,9582	0,0179
Precisão	0,9603	0,0143
Recall	0,9582	0,0179
F1-Score	0,9586	0,0167

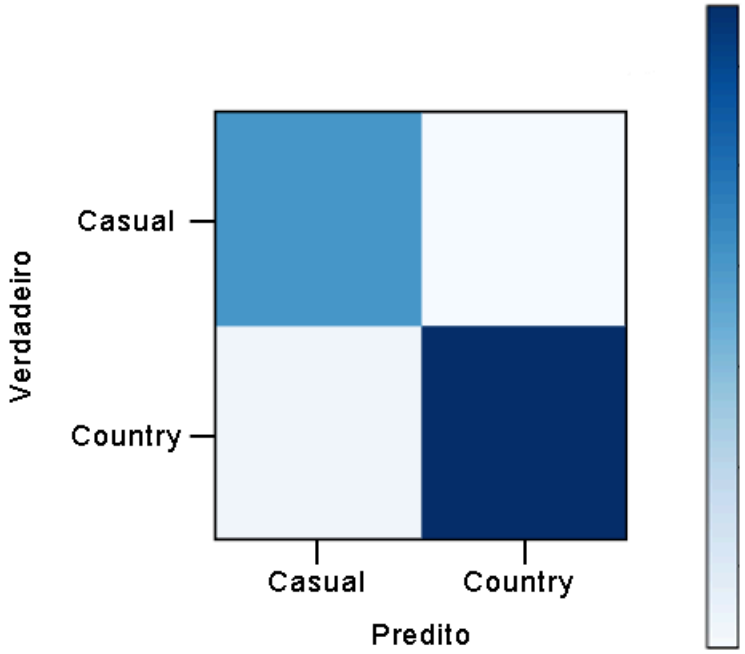


Valores médios das métricas do modelo CNN 1.

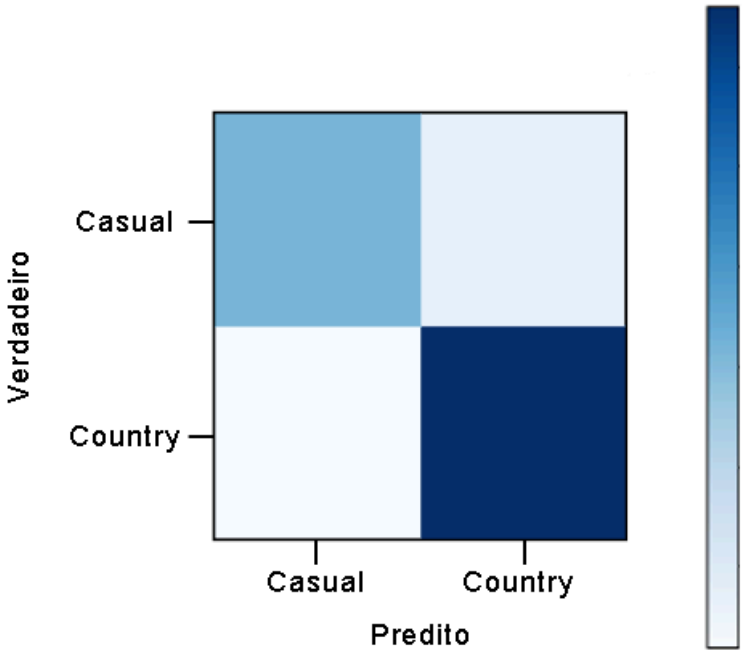
K=2		
Métrica	**Média**	**DP**
Acurácia	0,8655	0,1129
Precisão	0,8651	0,1145
Recall	0,8655	0,1129
F1-Score	0,8643	0,1136



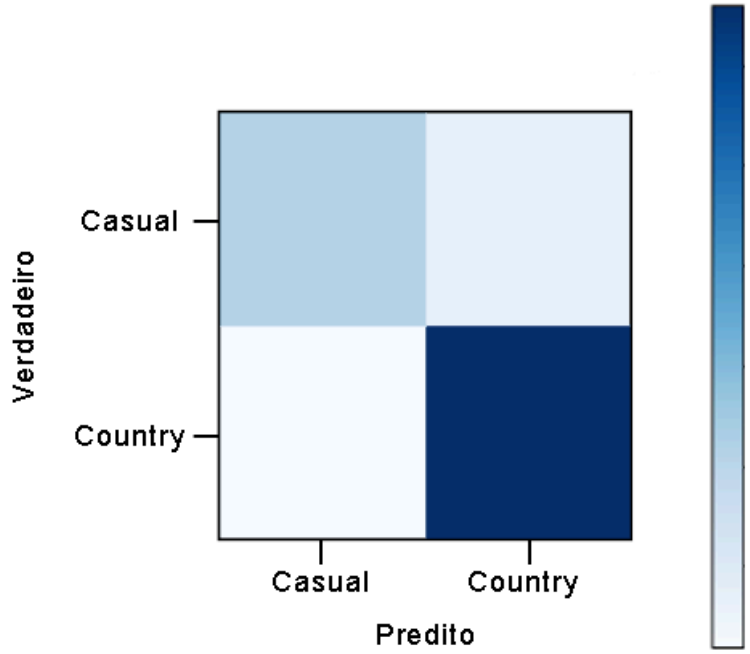
Fold 1:
Acurácia: 0.6727
Precisão: 0.6695
Recall: 0.6727
F1-score: 0.6702



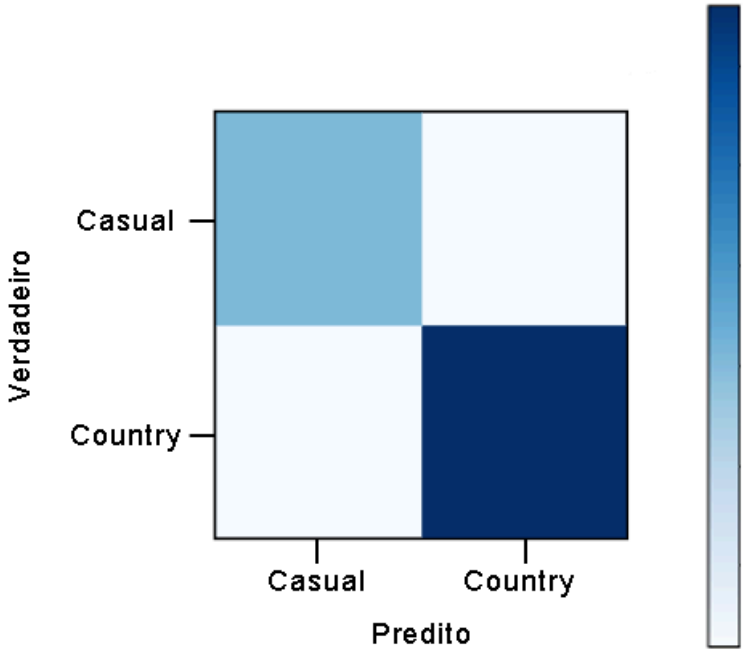
Fold 2:
Acurácia: 0.8455
Precisão: 0.8461
Recall: 0.8455
F1-score: 0.8457



Fold 3:
Acurácia: 0.9545
Precisão: 0.9575
Recall: 0.9545
F1-score: 0.9537



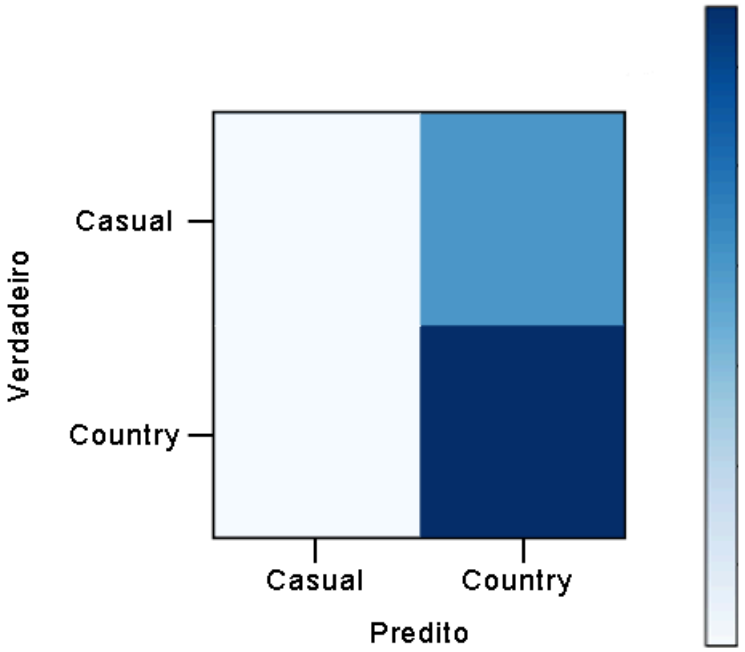
Fold 4:
Acurácia: 0.8545
Precisão: 0.8521
Recall: 0.8545
F1-score: 0.8520



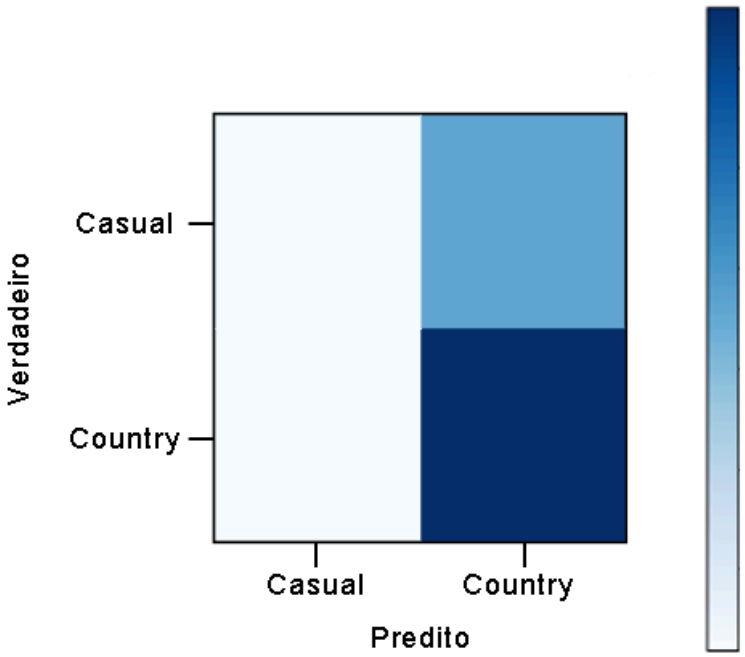
Fold 5:
Acurácia: 1.0
Precisão: 1.0
Recall: 1.0
F1-score: 1.0

Valores médios das métricas do modelo CNN 2

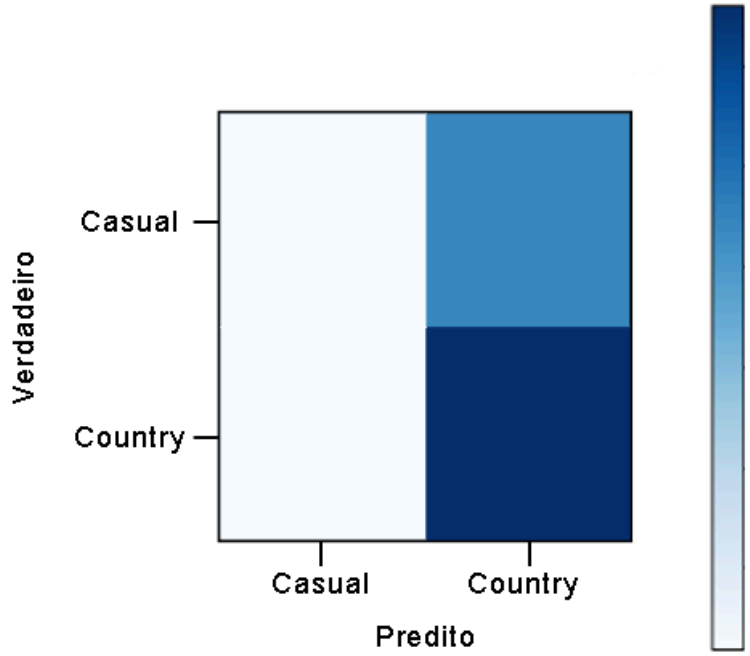
K=2		
Métrica	**Média**	**DP**
Acurácia	0,6400	0,0440
Precisão	0,4115	0,0580
Recall	0,6400	0,0440
F1-Score	0,5004	0,0560



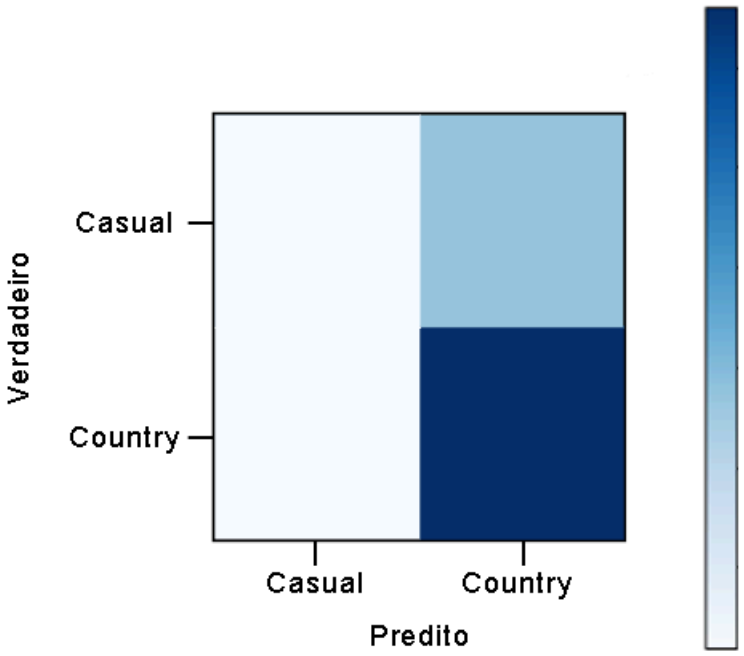
Fold 1:
Acurácia: 0.6273
Precisão: 0.3936
Recall: 0.6273
F1-score: 0.4836



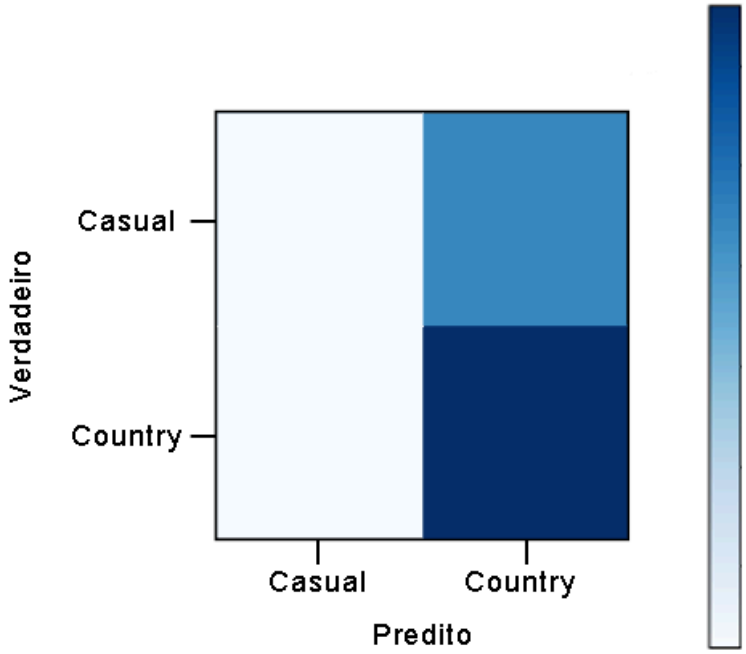
Fold 2:
Acurácia: 0.6545
Precisão: 0.4284
Recall: 0.6545
F1-score: 0.5179



Fold 3:
Acurácia: 0.6
Precisão: 0.36
Recall: 0.6
F1-score: 0.45



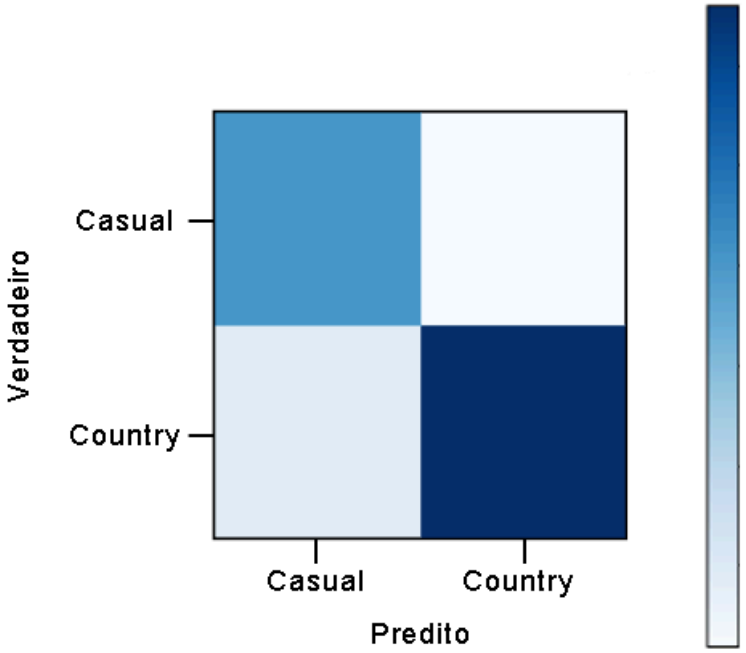
Fold 4:
Acurácia: 0.7182
Precisão: 0.5158
Recall: 0.7182
F1-score: 0.6004



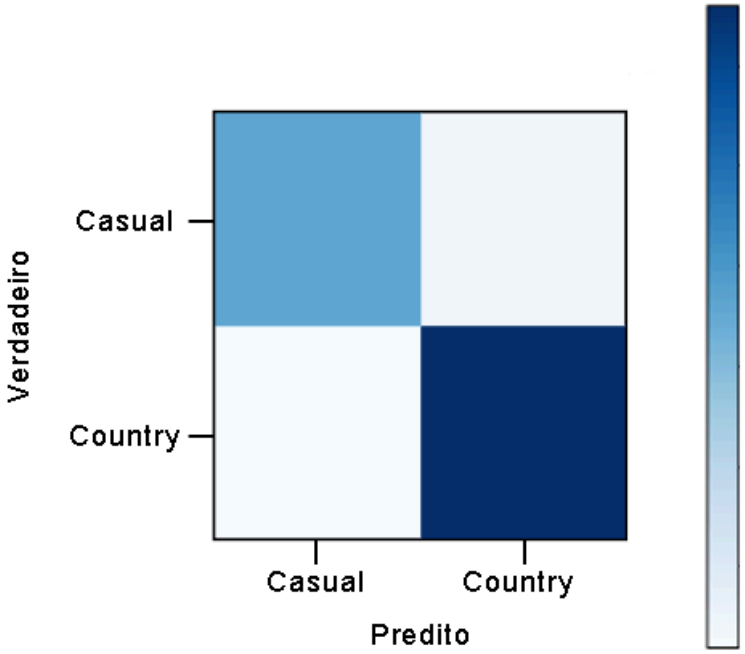
Fold 5:
Acurácia: 0.6
Precisão: 0.36
Recall: 0.6
F1-score: 0.45

Valores médios das métricas do modelo CNN 3

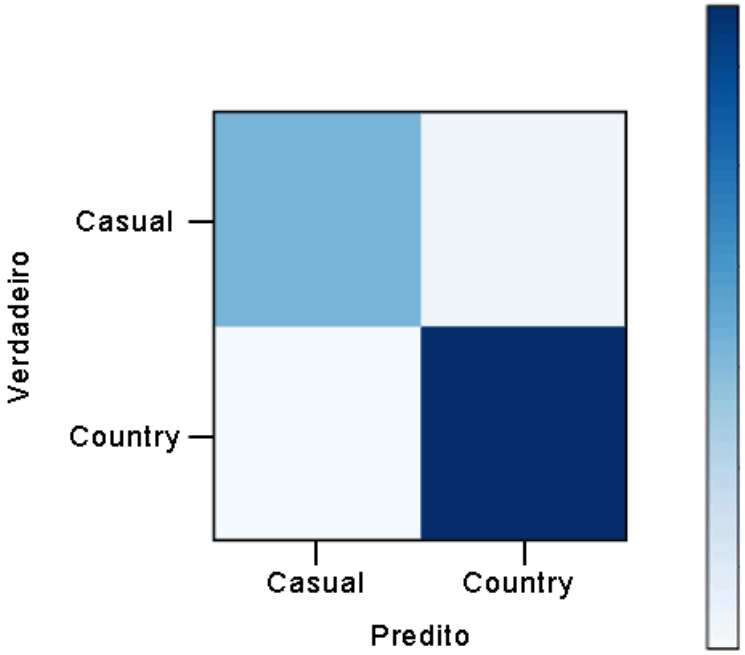
K=2		
Métrica	**Média**	**DP**
Acurácia	0,9545	0,0464
Precisão	0,9562	0,0442
Recall	0,9545	0,0464
F1-Score	0,9547	0,0458



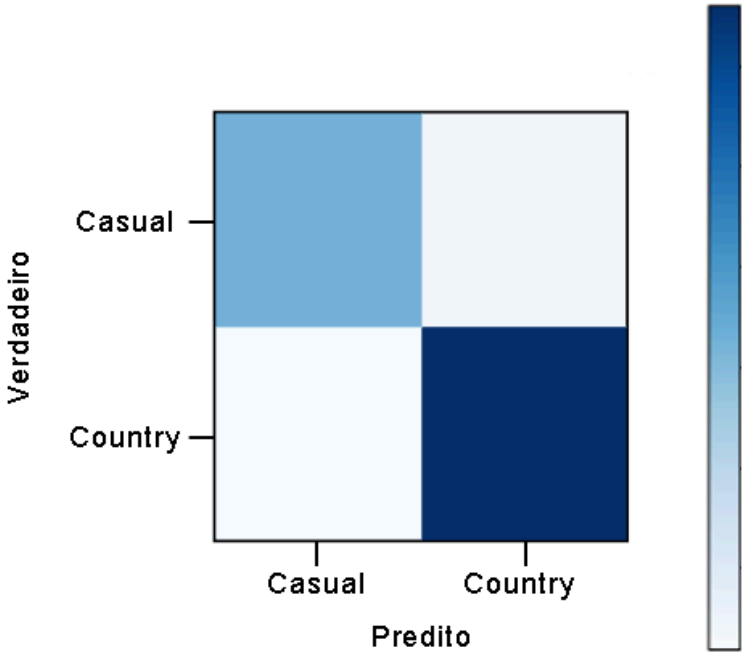
Fold 1:
Acurácia: 0.8636
Precisão: 0.8697
Recall: 0.8636
F1-score: 0.8649



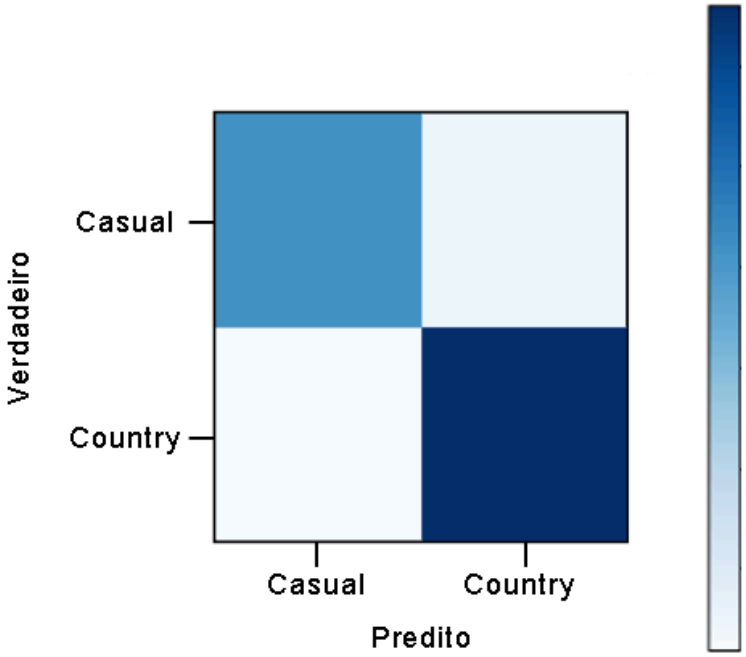
Fold 2:
Acurácia: 0.9636
Precisão: 0.9639
Recall: 0.9636
F1-score: 0.9634



Fold 3:
Acurácia: 0.9818
Precisão: 0.9823
Recall: 0.9818
F1-score: 0.9817



Fold 4:
Acurácia: 0.9909
Precisão: 0.9910
Recall: 0.9909
F1-score: 0.9909

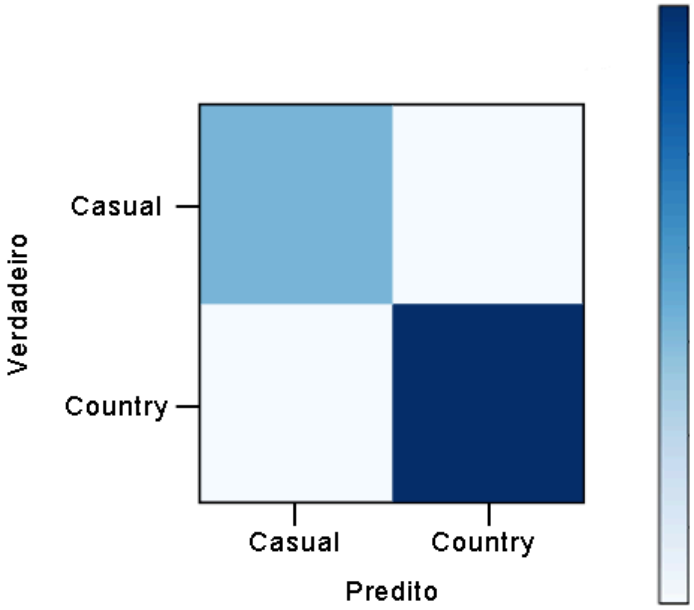


Fold 5:
Acurácia: 0.9727
Precisão: 0.9739
Recall: 0.9727
F1-score: 0.9725

APÊNDICE C - RESULTADOS PARA K = 7

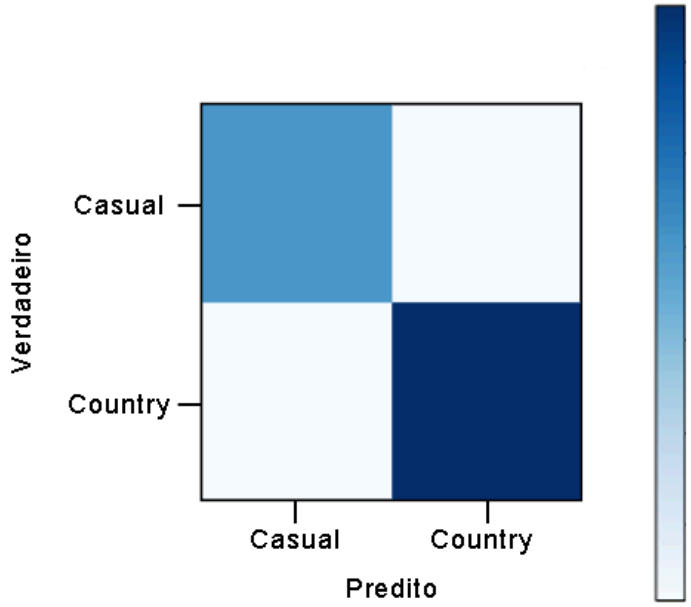
Valores médios das métricas do modelo Random Forest.

K=7		
Métrica	**Média**	**DP**
Acurácia	0,9296	0,0231
Precisão	0,9314	0,0245
Recall	0,9296	0,0231
F1-Score	0,9286	0,0242



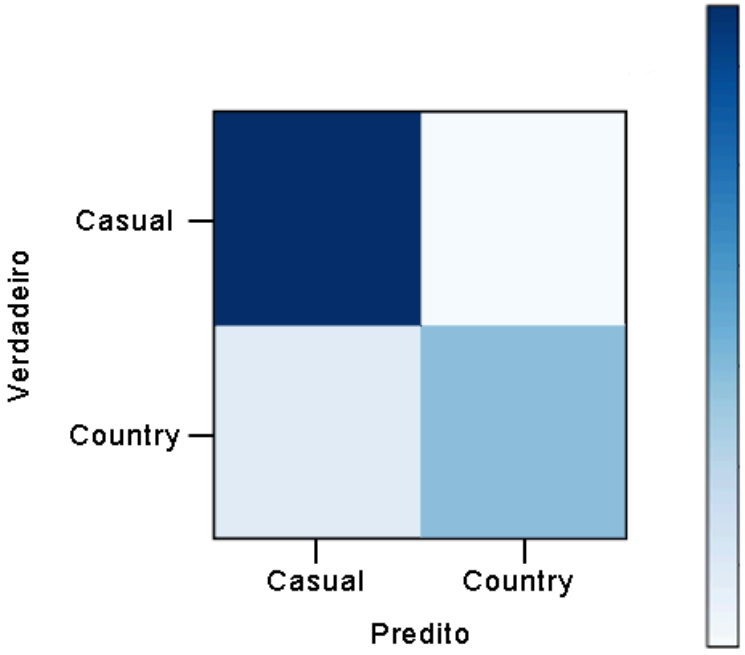
Valores médios das métricas do modelo SVM.

K=7		
Métrica	**Média**	**DP**
Acurácia	0,9630	0,0151
Precisão	0,9651	0,0136
Recall	0,9630	0,0151
F1-Score	0,9627	0,0156

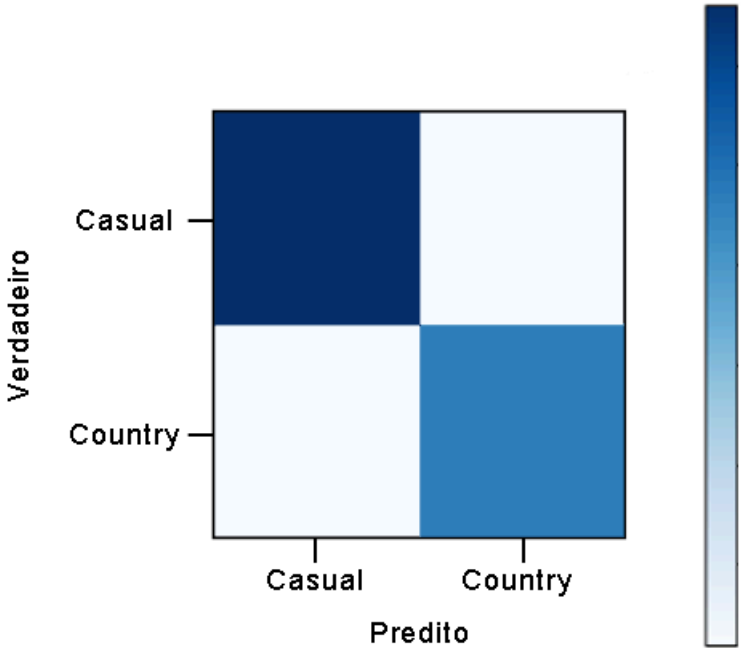


Valores médios das métricas do modelo CNN 1.

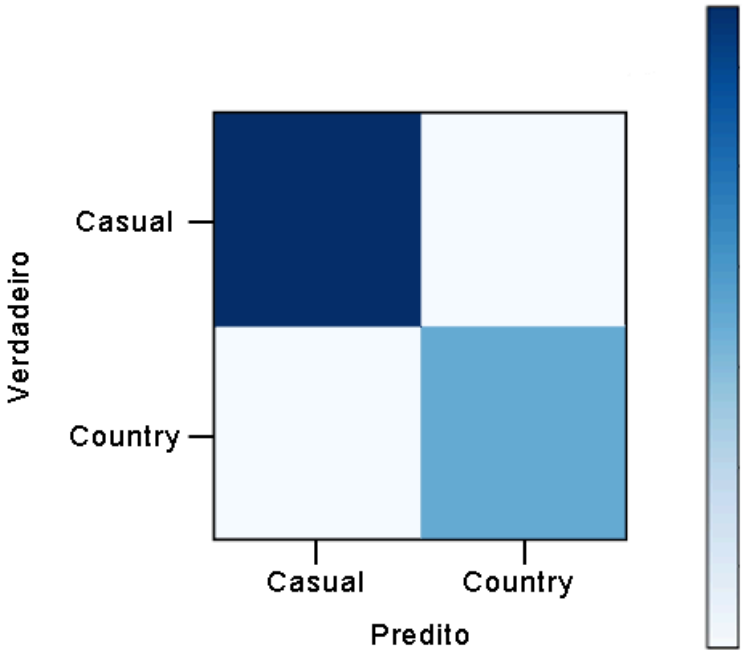
K=7		
Métrica	**Média**	**DP**
Acurácia	0,9673	0,0401
Precisão	0,9676	0,0396
Recall	0,9673	0,0401
F1-Score	0,9668	0,041



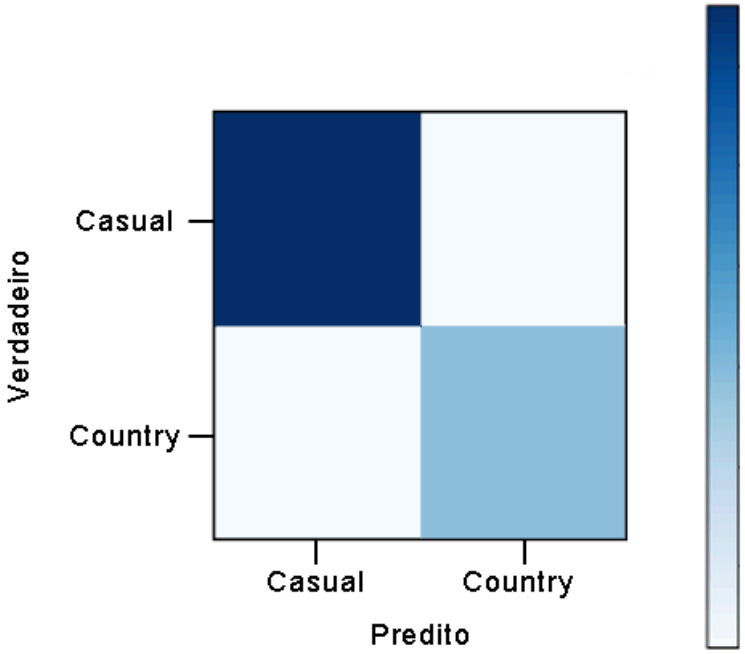
Fold 1:
Acurácia: 0.8909
Precisão: 0.8923
Recall: 0.8909
F1-score: 0.8886



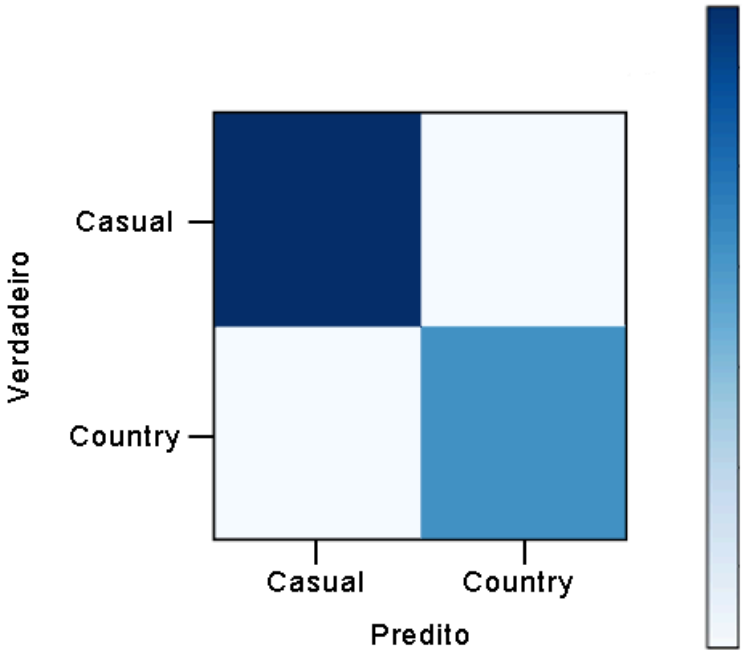
Fold 2:
Acurácia: 0.9636
Precisão: 0.9636
Recall: 0.9636
F1-score: 0.9636



Fold 3:
Acurácia: 0.9909
Precisão: 0.9910
Recall: 0.9909
F1-score: 0.9909



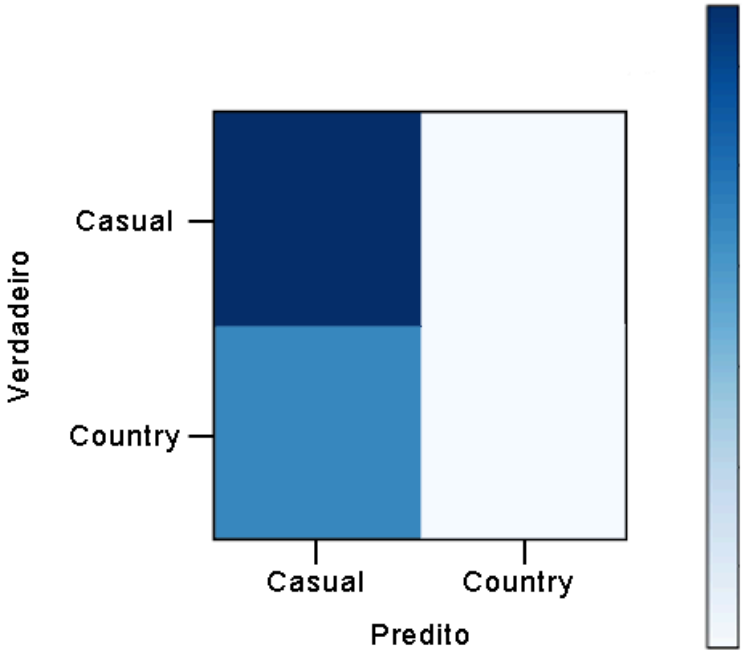
Fold 4:
Acurácia: 0.9909
Precisão: 0.9910
Recall: 0.9909
F1-score: 0.9909



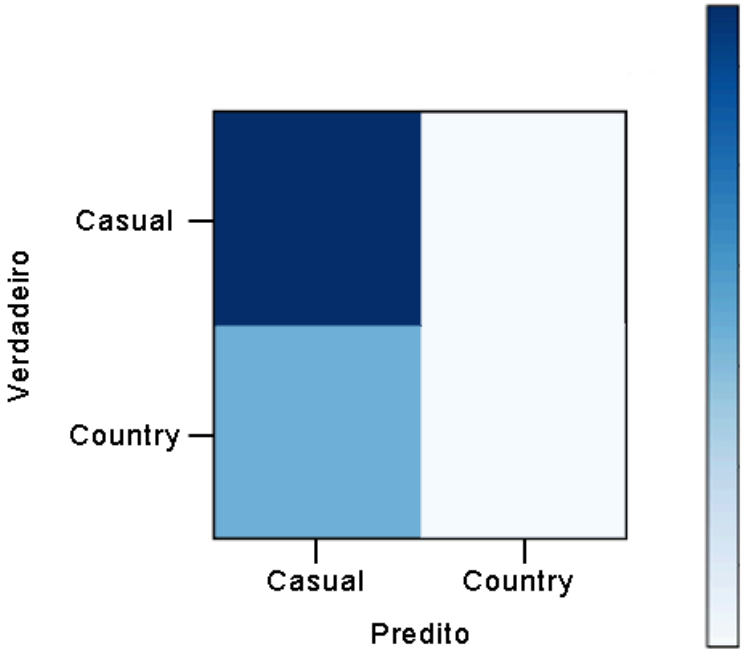
Fold 5:
Acurácia: 1.0
Precisão: 1.0
Recall: 1.0
F1-score: 1.0

Valores médios das métricas do modelo CNN 2.

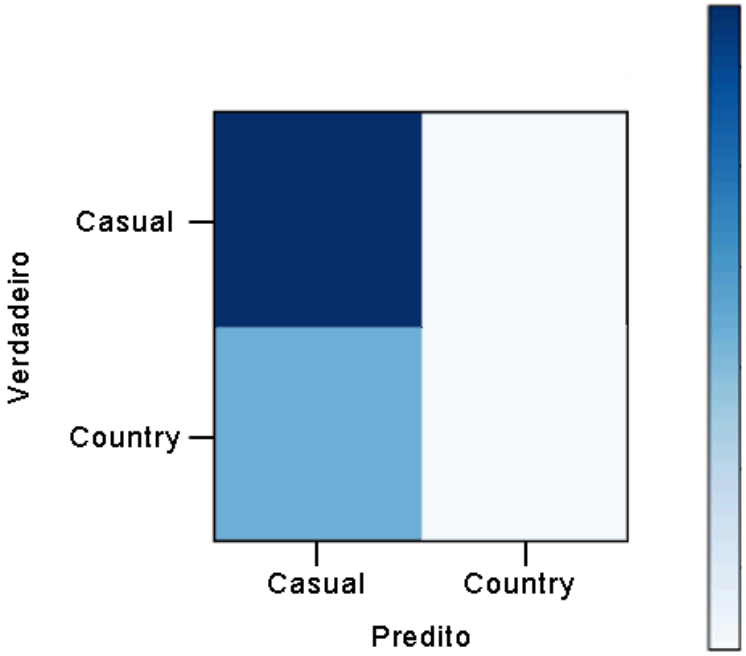
K=7		
Métrica	**Média**	**DP**
Acurácia	0,64	0,0405
Precisão	0,4112	0,0511
Recall	0,64	0,0405
F1-Score	0,5003	0,0505



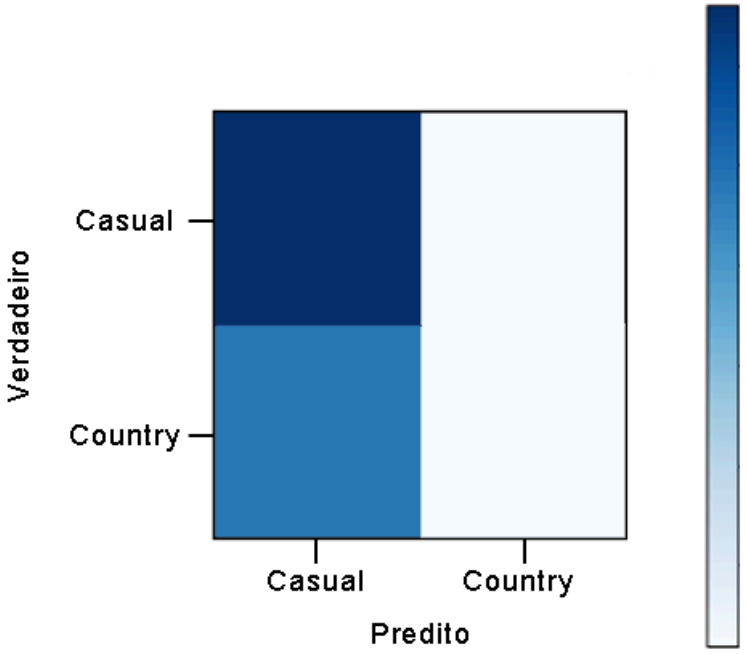
Fold 1:
Acurácia: 0.6
Precisão: 0.36
Recall: 0.6
F1-score: 0.45



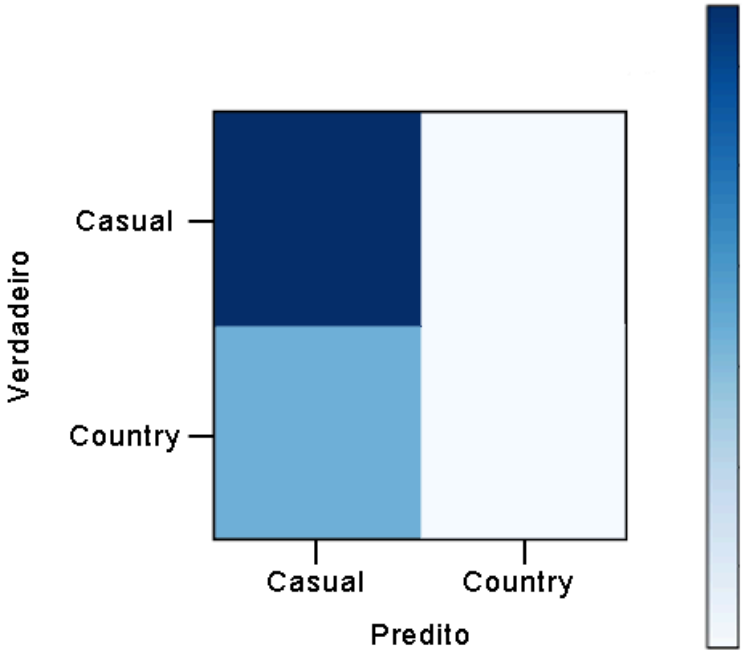
Fold 2:
Acurácia: 0.6727
Precisão: 0.4526
Recall: 0.6727
F1-score: 0.5411



Fold 3:
Acurácia: 0.6727
Precisão: 0.4526
Recall: 0.6727
F1-score: 0.5411



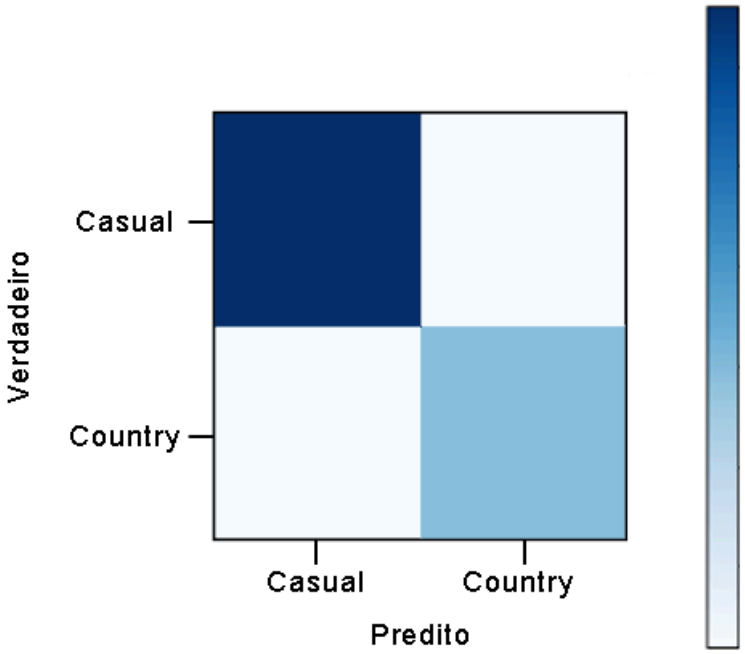
Fold 4:
Acurácia: 0.5818
Precisão: 0.3385
Recall: 0.5818
F1-score: 0.428



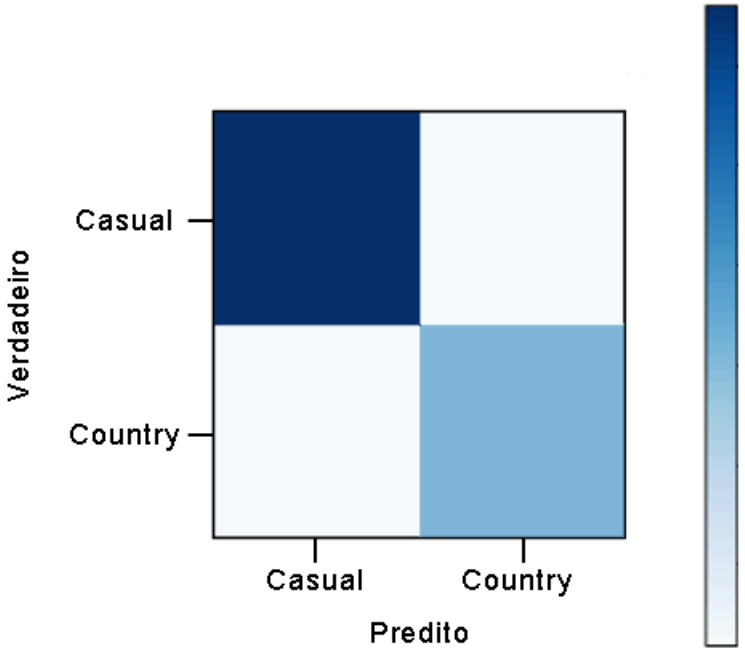
Fold 5:
Acurácia: 0.6727
Precisão: 0.4526
Recall: 0.6727
F1-score: 0.5411

Valores médios das métricas do modelo CNN 3.

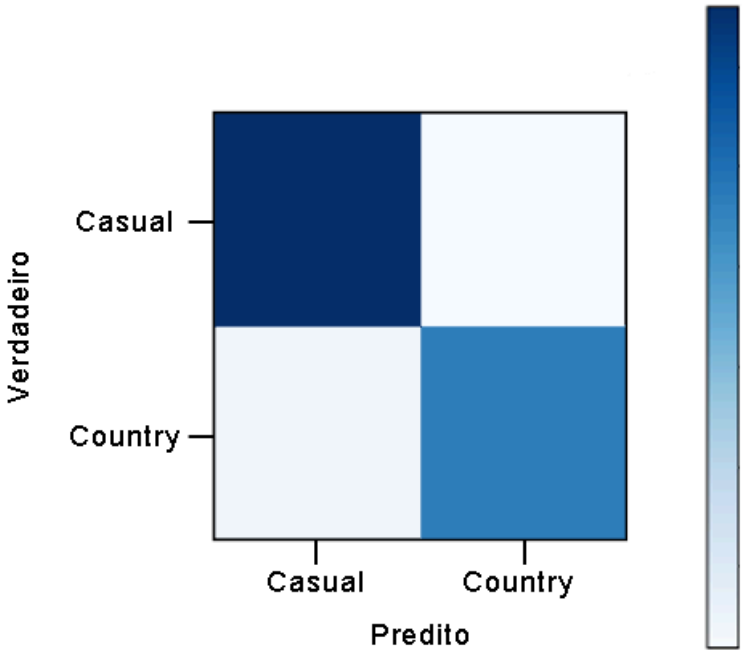
K=7		
Métrica	**Média**	**DP**
Acurácia	0,9691	0,0302
Precisão	0,9694	0,0303
Recall	0,9691	0,0302
F1-Score	0,9691	0,0302



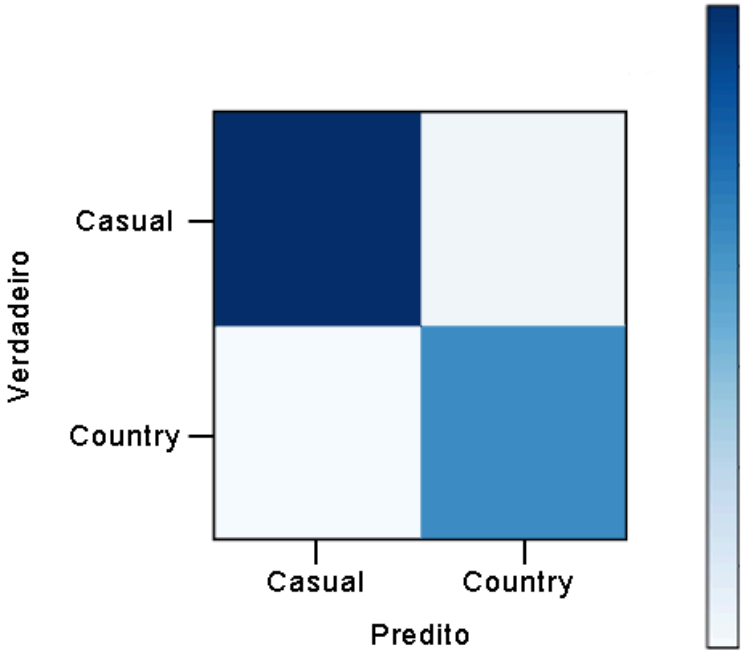
Fold 1:
Acurácia: 0.90909
Precisão: 0.9091
Recall: 0.9091
F1-score: 0.9091



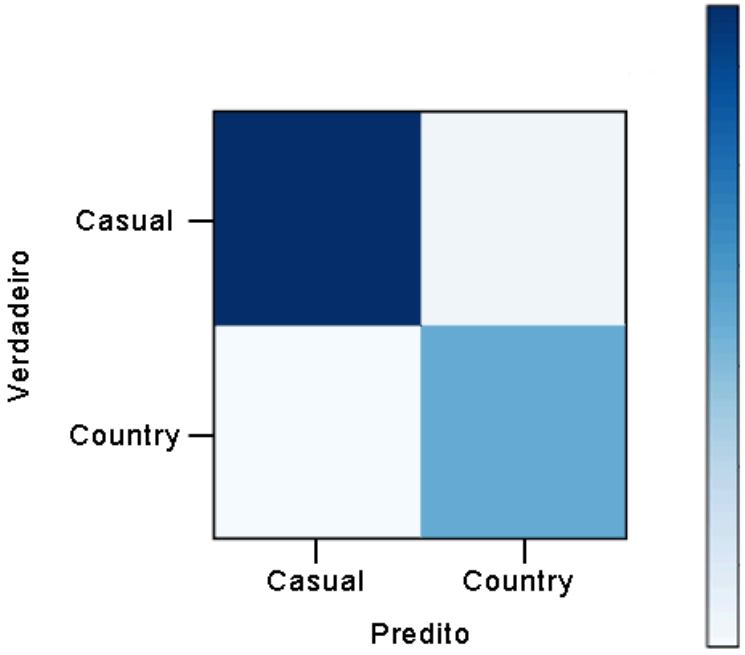
Fold 2:
Acurácia: 0.9818
Precisão: 0.9818
Recall: 0.9818
F1-score: 0.9818



Fold 3:
Acurácia: 0.9818
Precisão: 0.9823
Recall: 0.9818
F1-score: 0.9818



Fold 4:
Acurácia: 0.9818
Precisão: 0.9826
Recall: 0.981
F1-score: 0.9819



Fold 5:
Acurácia: 0.9909
Precisão: 0.9910
Recall: 0.9909
F1-score: 0.9909