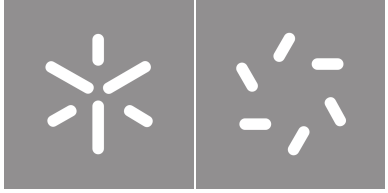
Universidade do Minho
Escola de Ciências

Rodrigo Filipe Rodrigues Rocha

Classificação de Preferências no Retalho Alimentar: uma Abordagem Supervisionada

Outubro de 2023

Classificação de Preferências no Retalho Alimentar: uma Abordagem Supervisionada
Rodrigo Rocha
UMinho | 2023



Universidade do Minho
Escola de Ciências

Rodrigo Filipe Rodrigues Rocha

Classificação de Preferências no Retalho Alimentar: uma Abordagem Supervisionada

Dissertação de Mestrado
Mestrado em Estatística para Ciência de Dados

Trabalho efetuado sob a orientação de:
Prof. Luís Meira Machado
Dra. Ana Freitas

Outubro de 2023

Agradecimentos

Os passados 5 anos foram anos repletos de desafios e conquistas que me fizeram crescer, quer como profissional quer como pessoa. É importante reforçar que estas conquistas seriam impossíveis de alcançar sem o apoio e motivação das pessoas que me rodeiam e por isso dedico esta secção a eles.

Queria começar por agradecer aos meus pais, Dona Modesta e Sr. Rodrigo, por tudo o que fizeram por mim durante toda a minha vida e por serem responsáveis de grande parte da pessoa que sou hoje. Aos meus avós, Dona Sãozinha e Sr. Manel pela maravilhosa comida que me permitem consumir sempre que preciso e por todos os ensinamentos que, à sua maneira, me passaram. Aos meus tios, Sr. Joel e Dona Micas, por terem tido paciência para me aturar em criança, por continuarem a ter agora em adulto e por terem tido um papel fulcral na minha educação e desenvolvimento. À Balbina Lisboa por todo o apoio que me deu nas diferentes fases da minha vida e por ter sido a minha maior conselheira no processo de escolha de curso sufperior. E por fim à restante família que sempre deu o seu melhor para fazer de mim uma melhor pessoa.

Aos 007 (Castro, Maia, Gonçalves, Dino, Vitor e Tiago) por me terem acompanhado ao longo de todo o meu percurso académico, não só como colegas de curso mas também como amigos com os quais pude contar sempre que precisei. Ao Jhonathan Barrios por todas as dicas que me deu ao longo do mestrado e por sempre se ter preocupado comigo. És uma máquina de vencer. Ao Sr. Carlos Sousa, parceiro estagiário da Sonae, pessoa com quem partilhei grande parte do período da dissertação. Por todos os conselhos, piadas e factos aleatórios que tornaram esta experiência muito mais fácil. Aos OG's (João Mendes, Bia, Pedro, Catarina, Bruno, Leandro, Duarte, João, Diogo e Rafael Saraiva, pessoas que estão presentes na minha vida há longos anos e que espero que possam continuar a estar muitos mais!

À restante equipa da Sonae (Catarinha Pinho, Catarina Almeida, Filipe, Alexandre, Ana Carvalho e Liliana Bernardino) por me terem permitido ser eu mesmo, ajudado sempre que tinha uma dúvida e pelos grandes momentos de "galhofa" partilhados nos almoços de terça.

Aos meus orientadores, Dra. Ana Freitas e Prof. Luís Filipe, pela disponibilidade que sempre tiveram para reunir e esclarecer dúvidas, pelos ensinamentos que me foram passando ao longo da dissertação que certamente vão fazer de mim uma melhor pessoa e profissional.

Por fim, quero agradecer à minha querida namorada, Renata Martins, pessoa com quem partilho todos os meus dias. Por me incentivar sempre a ir mais além e nunca duvidar de mim. Por todas as vezes que me deu na cabeça para não procrastinar, por lidar comigo nos piores momentos sempre com uma imensa positividade e por, indiretamente, ser para mim um exemplo de superação e persistência.

Declaração de Integridade

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do MinhoUniversidade do Minho.

Resumo

Classificação de Preferências no Retalho Alimentar: uma Abordagem Supervisionada

Com a atual exigência por parte dos consumidores, os retalhistas são obrigados a estar ao nível dos mesmos e sendo o Continente um retalhista que aposta na personalização do serviço prestado, não é exceção. Desta forma, dado que o Continente comunica com os seus clientes por diversos canais (Folheto personalizado, cupões, etc.), é necessários que estes contenham o máximo de personalização possível no que toca a exigências/restrições alimentares. Tendo isto como premissa, o projeto baseia-se no desenvolvimento de um sistema de classificação automático de produtos em diferentes categorias (vegetariano, *vegan*, sem lactose, sem glúten, sem açúcar e biológico) utilizando diferentes abordagens.

Este projeto une diferentes abordagens de *Machine Learning* e *Deep Learning* para responder ao problema proposto. O projeto está dividido em três grandes partes, classificação de imagem, classificação texto e *ensembles*, sendo este uma junção das duas anteriores e uma "quarta" parte de extração de texto de imagem.

Para a classificação de imagem, foram testadas diferentes técnicas, sendo que inicialmente foram utilizadas duas arquiteturas pré-treinadas, *EfficientNet* e VGG-16, que foram posteriormente treinadas num *dataset* obtido com imagens de produtos do Continente *Online* e utilizadas como classificadoras. Foi utilizada uma segunda abordagem onde as mesmas arquiteturas foram utilizadas como extratoras de características das imagens e posteriormente alimentadas a um *support vector machine* para classificação.

Na classificação de texto foram testadas também duas abordagens, primeiramente foi criada uma rede neuronal convolucional e utilizada uma matriz de *embeddings* pré-treinada. Numa segunda abordagem foi utilizada a arquitetura pré-treinada *Bidirecional Encoder Representations from Transformers* como classificadora, treinada num *dataset* com descrições de produtos.

No *ensemble* foi utilizada uma regressão logística alimentada com as probabilidades de classificação do melhor modelo de texto e imagem.

Para finalizar tentou-se implementar uma "quarta" parte que iria servir como complemento ao modelo de classificação de texto, que seria a extração de texto de uma imagem. Foi utilizada uma tecnologia de reconhecimento ótico de caracteres.

Palavras-chave: *Support Vector Machine*, *Machine Learning*, *Deep Learning*, Classificação Automática, *Embeddings*

Abstract

Preference Ranking in Food Retail: A Supervised Approach

With today's consumer demands, retailers are obliged to keep up with them, and Continente a retailer that invests heavily in personalizing the service provided, is no exception. Therefore, given that Continente communicates with its customers through various channels (personalized leaflets, coupons, etc.), it is necessary that these contain as much personalization as possible when it comes to dietary requirements/restrictions. With this as a premise, the project is based on developing a system for automatically classifying products into different categories (vegetarian, vegan, lactose-free, gluten-free, sugar-free and organic) using different approaches.

This project brings together different approaches from Machine Learning and Deep Learning to answer the proposed problem. The project is divided into three main parts, image classification, text and ensembles, the latter being a combination of the previous two and a "fourth" part for extracting text from images.

For the first part, image classification, different techniques were tested, initially using two pre-trained architectures, EfficientNet and VGG-16, which were then trained on a dataset obtained with product images from Continente Online and used as classifiers. Since the results were unsatisfactory, a second approach was used where the same architectures were used as feature extractors from the images and then fed to a Support Vector Machinefor classification.

In text classification, two approaches were also tested: first, a Convolutional Neural Network was created and a pre-trained nembeddings matrix was used. A second approach used the pre-trained Bidirectional Encoder Representations from Transformers architecture as a classifier, trained on a dataset with product descriptions.

In the ensemble, a logistic regression was used, fed with the classification probabilities of the best text and image model.

Finally, an attempt was made to implement a "fourth" part that would serve as a complement to the text classification model, which would be the extraction of text from an image. Optical character recognition technology was used.

Keywords: Support Vector Machine, Machine Learning, Deep Learning, Automatic Classification, Embeddings,

Índice

Índice de Figuras

Índice de Tabelas

Índice de Listagens

Siglas

AI	Artificial Intelligence
ANN	Artificial Neural Network
BERT	Bidirectional Encoder Representations from Transformers
BI	Business Intelligence
CAPTCHA	Completely Automated Public Turing Test To Tell Computers and Humans Apart
Cbow	Continuous bag-of-words
CNN	Convolutional Neural Network
DDoS	Distributed Denial of Service
DL	Deep Learning
GloVe	Global Vectors for Word Representation
GPT	Generative Pre-trained transformer
ML	Machine Learning
NLP	Natural Language Processing
OCR	Optical Character Recognition
ReLU	Rectified Linear Unit
SGD	Stochastic Gradient Descent
SKU	Stock Keeping Unit
SMOTE	Synthetic Minority Over-sampling Technique
SQL	Structured Query Language

SSD Single Shot MultiBox Detector

SVM Support Vector Machine

VGG Visual Geometry Group

YOLO You Only Look Once

Capítulo 1

Introdução

1.1 Contextualização

Esta secção apresenta a MC, empresa onde este projeto foi desenvolvido, bem como a motivação para a escolha/desenvolvimento do mesmo. Serão também apresentados os objetivos principais.

1.1.1 MC Sonae

A MC é uma empresa multinacional que gere diversos de negócios em diferentes sectores, tais como retalho e bem-estar (MC Sonae), serviços financeiros (Universo e Bright-Pixel), tecnologia (Worten), imobiliário (Sierra), telecomunicações (NOS), moda (Zeitreel), desporto (ISRG) e ingredientes à base de plantas (Sparkfood Sonae). A MC é a principal empresa de retalho alimentar em Portugal, Com várias áreas de negócio distintas como o Continente (hipermercados), Continente Modelo (supermercados) e Continente Bom Dia (supermercados de conveniência), Meu Super (loja de proximidade em franchising), Bagga (cafetaria), Go Natural (supermercados e restaurantes orientados para o saudável e sustentável), Make Notes e Note! (livraria e papelaria), ZU (produtos e serviços para animais de estimação), Well's (saúde, bem-estar e óptica), Dr.Wells (odontologia e medicina cosmética e beleza), Zippy e Mo (têxteis para crianças e adultos) e Cozinha Continente (restaurante).

1.2 Motivação

O Artificial Intelligence (AI) tem sido uma força que impulsiona a inovação em diversos setores da sociedade, desde *chatbots* de negócios que permitem auxiliar o atendimento ao cliente, até carros com capacidade de dirigirem autonomamente pelas estradas. No centro desta inovação está o Deep Learning

(DL), um sub-campo do *Machine Learning* que tem demonstrado uma elevada capacidade de compreensão e interpretação de dados complexos.

Ao passo que o AI continua a progredir, a sua capacidade de interagir e entender informações no formato de imagem e texto é cada vez mais crucial para análises de dados médicos, deteção de fraudes financeiras e melhoria da experiência do cliente. Este projeto permitiu mergulhar no mundo do DL e a exploração de diferentes técnicas de classificação de imagem e texto. Outra grande motivação deste projeto foi elevar o nível da comunicação personalizada da empresa, bem como a necessidade de respeitar as preferências alimentares do cliente.

1.3 Objetivos

O objetivo deste estágio é a classificação dos produtos do hipermercado Continente através das suas características. Em adição à separação entre vegetais, laticínios, carne e peixe, é necessário criar também uma partição de produtos sem glúten, *vegan*, sem lactose, etc. Para isso será necessário recorrer a abordagens supervisionadas de *Machine Learning* e técnicas de *Natural Language Processing*.

Inicialmente será realizada uma introdução ao *Python*, linguagem que será utilizada durante o estágio, de forma a aplicar e ampliar o conhecimento nesta linguagem. Para isso serão testados alguns algoritmos de reconhecimento de imagem, onde o propósito será identificar objetos presentes em fotos ou vídeos.

De seguida, será feita uma análise inicial aos dados que a empresa fornecerá, de maneira a que consiga ter uma melhor noção sobre a forma que estes têm, informação presente nos mesmos, etc. Após esta análise, serão explorados e testados diversos algoritmos posteriormente decidir quais os mais adequados a utilizar.

Posteriormente, será então implementado e testado o algoritmo nos dados fornecidos, feita uma análise da eficácia do mesmo e efetuadas alterações de forma a que se possa maximizar a eficiência.

Capítulo 2

Estado de Arte

Nesta secção serão abordados vários conteúdos que foram necessários no desenvolvimento deste projeto. Serão tratados vários campos do AI associados à classificação de imagem e extração de texto.

2.1 Aprendizagem automática (*Machine Learning*)

Machine Learning (ML) pode ser explicado como uma forma de automatizar e melhorar o processo de aprendizagem do computador sem a assistência humana. É um ramo da Artificial Intelligence (AI) que sintetiza as relações entre os dados e a informação [1]. O processo inicia-se com a introdução de dados de boa qualidade num modelo, posteriormente treinado com base nos dados fornecidos. Na Figura 2 está presente um diagrama representativo das diferenças entre a programação clássica e ML.

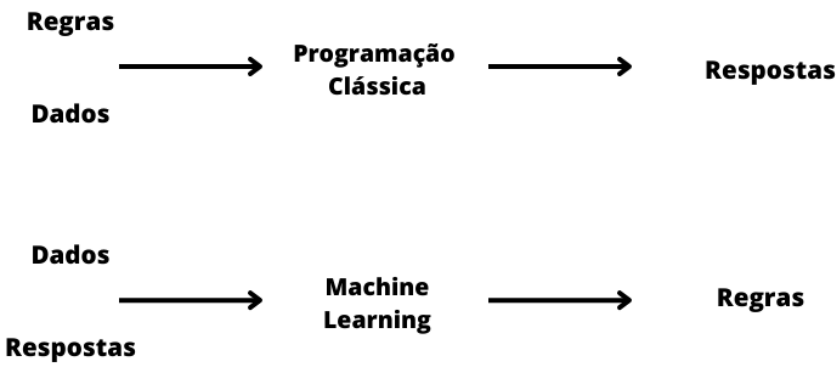


Figura 1: Esquema ML

Ainda que o ML apenas tenha começado a florescer nos anos 90 [2], rapidamente se tornou o ramo do AImais popular e nos dias de hoje está presente em diversos contextos, por exemplo nos anúncios que

aparecem quando se abre um *website*, mercados financeiros, previsão meteorológica, cupões promocionais segmentados, etc.

A crescente fama que o ML tem vindo a ganhar deve-se à capacidade deste estabelecer relações implícitas dentro de grandes conjuntos de dados de forma a resolver problemas, reconhecer padrões, etc.

Para a utilização de ML subentende-se a necessidade de três peças indispensáveis:

- Dados de entrada - Informação inicial, dependendo do objetivo esta poderá ser ficheiros de som, imagem, coordenadas de dados, etc.
- Resultados esperados - Informação dos *outputs* expectáveis
- Uma forma de medir a qualidade do modelo - Muitas vezes conhecida como *Loss Function*, serve para fazer comparação entre o *output* obtido pelo modelo e o *output* esperado. Este passo é o que chamamos de aprendizagem (*learning*)

Há atualmente três abordagens básicas do ML: Aprendizagem supervisionada, aprendizagem não supervisionada e aprendizagem de reforço. A escolha da abordagem depende do tipo de problema que é proposto.

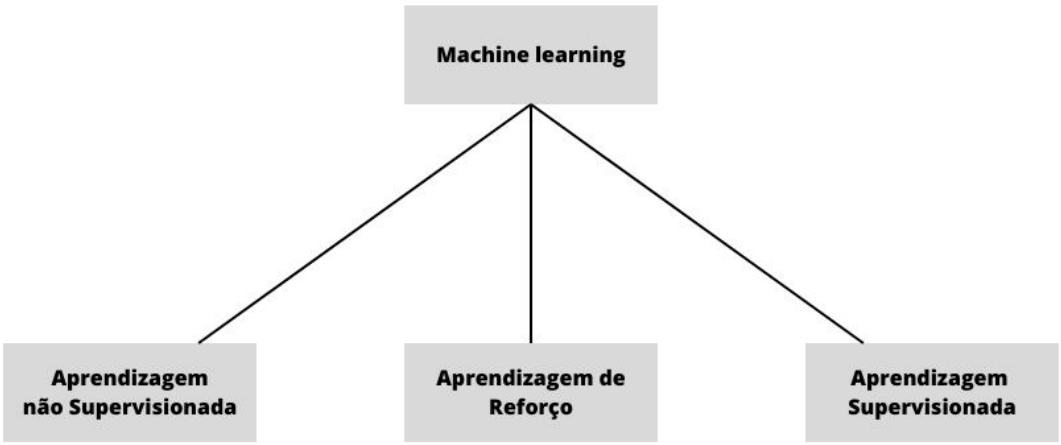


Figura 2: Tipos de abordagens de ML

2.1.1 Aprendizagem supervisionada

Nesta abordagem, as entradas são inseridas no modelo juntamente com as respostas desejadas para cada nó na camada de saída. O modelo processa a entrada e produz uma saída e as diferenças

entre a saída desejada e a saída real são então calculadas e usadas para ajustar os pesos do modelo de acordo com regras de aprendizagem específica, como o algoritmo de *backpropagation* ou o *perceptron*. O termo "supervisionado" refere-se ao facto da saída desejada ser fornecida por um "professor" externo, por outras palavras, o utilizador [3].

Este tipo de abordagem é bastante utilizado em dois tipos de problemas, problemas de classificação, como é o caso de árvores de decisão, Support Vector Machine (SVM) e problemas de regressão, como por exemplo regressão linear e regressão logística.

2.1.1.1 Regressão logística

Os problemas de regressão logística são usados quando é necessário a divisão das observações em duas categorias como por exemplo detetar se uma pessoa sofre ou não de uma determinada doença ou se um produto pertence ou não uma determinada categoria. Esta abordagem tem como objetivo encontrar relações entre uma ou várias variáveis preditoras e uma variável de resposta.

2.1.1.2 Regressão linear simples

A regressão linear simples é uma técnica que procura determinar relações entre variáveis (características) independentes e uma variável (resposta) dependente. É utilizada como um método de modelação preditiva para prever resultados contínuos. Alguns exemplos desta técnica são estudos para perceber se há relação entre o número de dias por semana que há a prática de desporto e o nível de stress e relação entre horas de estudo e uma nota num exame.

2.1.1.3 Regressão linear múltipla

Por vezes modelos de regressão linear simples têm limitações, isto é, quando há uma necessidade de analisar a influência de várias variáveis independentes sobre uma outra variável dependente [4]. Nesta análise, a variável resposta é a que desejamos prever ou explicar, enquanto que as variáveis preditoras são aquelas que acreditamos ter algum efeito na variável resposta. Cada variável preditora tem o seu próprio coeficiente que representa o efeito esperado na variável resposta quando há um aumento, ou diminuição, de uma unidade nessa variável preditora.

Uma forma de aplicação desta técnica é a previsão do preço de uma casa com base em diversas variáveis (idade da casa, localidade, número de quartos, etc.).

2.1.1.4 *Support Vector Machine*

SVM são algoritmos para a aprendizagem de regras através de dados provenientes de problemas de regressão e classificação. Foram sugeridos em meados dos anos 60 por Vapnik para problemas de classificação [5]. De uma forma muito breve, o objetivo do SVM para classificação binária é encontrar o

hiperplano ideal que divida os dados em duas classes. Como ideal define-se o hiperplano que maximiza a distância entre os dois pontos mais próximos das duas classes[6].

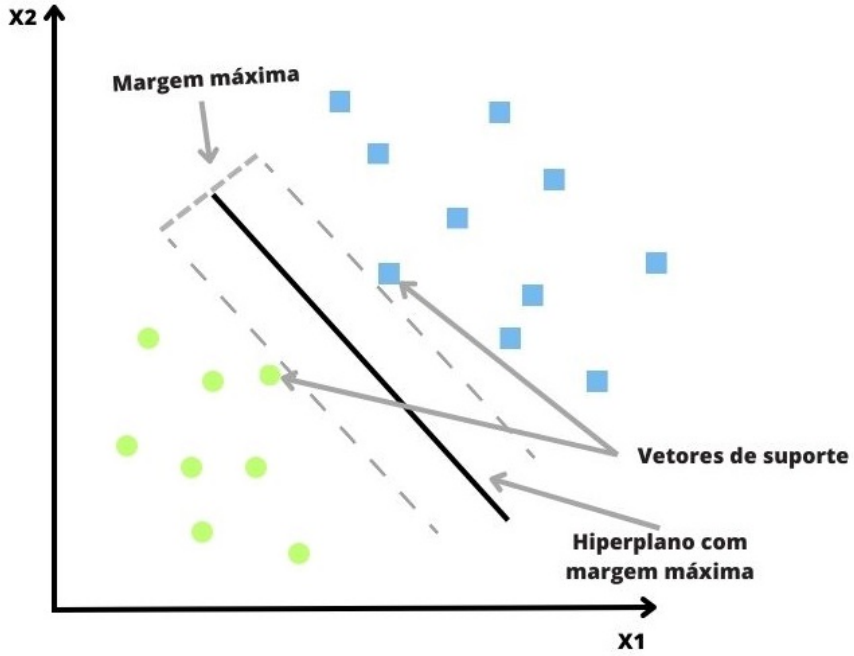


Figura 3: Algoritmo SVM

A forma mais simples para dividir duas categorias é com uma linha reta (ver Figura 3), porém nem sempre é possível. Dessa forma, quando não é possível dividir os dados com apenas uma linha, o SVM utiliza um *kernel* para mapear os dados para uma dimensão superior onde é possível fazer essa divisão, como é possível ver na Figura 4 [7].

Por natureza os SVM são classificadores binários. Ainda assim, existem estratégias que podem ser utilizadas de forma a adaptar estes algoritmos para problemas de classificação com múltiplas classes, duas das mais conhecidas são *One-Against-One (1A1)* e *One-Against-All (1AA)*. [8]

2.1.2 Abordagem não supervisionada

Na aprendizagem não supervisionada, o algoritmo está encarregue de descobrir as características estatisticamente relevantes dos dados de entrada. Ao contrário da aprendizagem supervisionada, não há um conjunto pré-definido de categorias onde os padrões devem ser classificados, em vez disso, o algoritmo deve procurar padrões e classificar os dados em grupos mais pequenos com base nesses mesmos padrões.

Há vários métodos que estão contidos dentro desta abordagem, tais como o *clustering* e a análise de componentes principais.

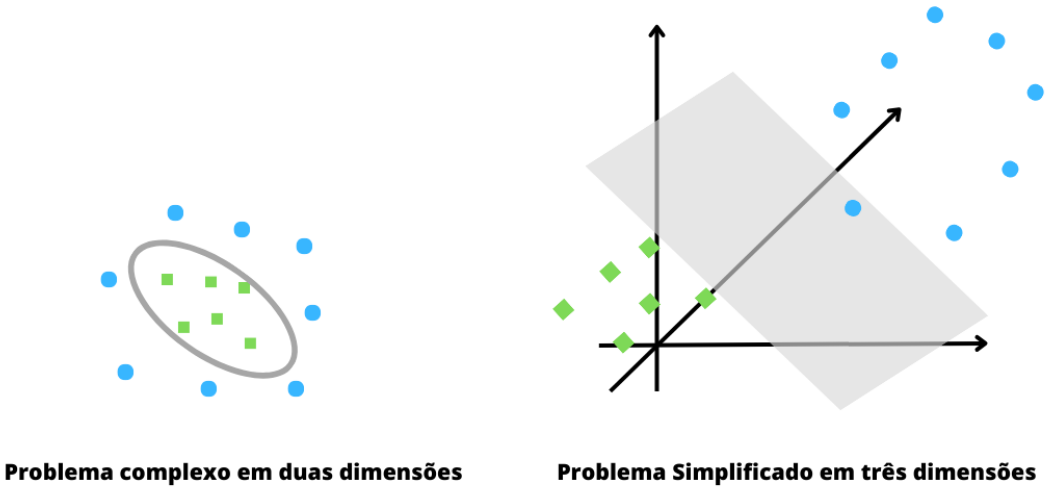


Figura 4: Mapeamento dos dados para uma dimensão superior

2.1.3 Aprendizagem de reforço (*Reinforcement learning*)

A aprendizagem de reforço é um processo de aprendizagem onde o objetivo é maximizar uma recompensa numérica ao tomar ações em diferentes situações. Ao contrário da maioria das formas de ML, o "aprendiz" não é fornecido com informações sobre quais ações devem ser tomadas; em vez disso, ele deve descobrir qual ação produz a maior recompensa através da experimentação. Em casos desafiadores, as ações podem afetar não apenas a recompensa imediata, mas também a situação futura e todas as recompensas subsequentes. As duas principais características da aprendizagem de reforço são a procura por erro e recompensa a longo prazo.

Existem atualmente diversos métodos e algoritmos de aprendizagem de reforço, como por exemplo o *Q-learning* e o *SARSA (State Action Reward State Action)*.

2.2 Aprendizagem Profunda (Deep Learning)

Deep Learning (DL) é um subcampo específico do ML: uma abordagem para aprender representações a partir de dados que enfatiza a aprendizagem de camadas sucessivas de representações cada vez mais significativas. O "profundo" em DL não se refere a qualquer tipo de compreensão mais profunda alcançada pela abordagem, mas sim à ideia de camadas sucessivas de representações. O número de camadas que contribuem para um modelo de dados é chamado de profundidade do modelo. O DL moderno muitas vezes envolve dezenas ou até centenas de camadas sucessivas de representação, todas aprendidas automaticamente a partir da exposição aos dados de treino. Enquanto isso, outras abordagens de ML tendem a concentrar-se apenas numa ou duas camadas de representação dos dados, daí o termo

aprendizagem superficial (*shallow learning*)) Em DL, essas representações em camadas são, quase sempre, aprendidas através de modelos chamados redes neuronais, estruturadas em camadas "empilhadas" umas nas outras.

O termo rede neuronal é uma referência à neurobiologia, mas embora alguns dos conceitos centrais em DL tenham sido desenvolvidos com inspiração na compreensão do cérebro não há evidências de que o cérebro implemente algo semelhante aos mecanismos de aprendizagem usados nos modelos modernos de DL.

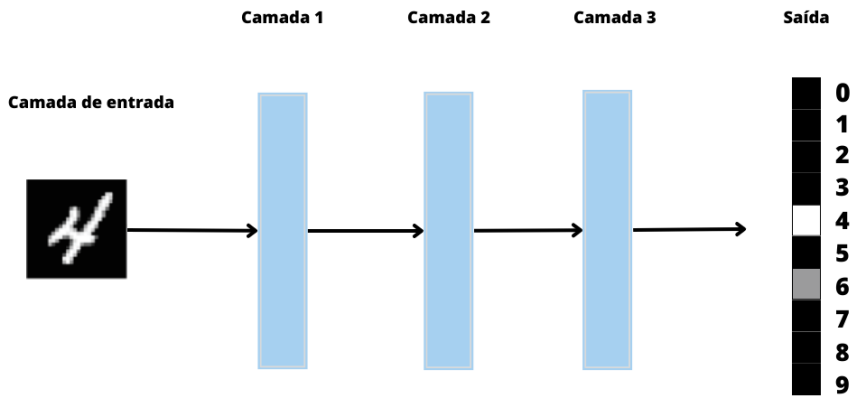


Figura 5: Rede Neuronal profunda para classificação de números

Na Figura 6, o objetivo é mapear entradas (como imagens de números) para alvos (como a *label* "4"), o que é feito analisando vários exemplos de entrada e resultados esperados. Sabe-se também que as redes neuronais profundas fazem esse mapeamento de entrada-resultado esperado através de uma sequência profunda de transformações de dados simples (camadas) e que essas transformações de dados são aprendidas por exposição a exemplos. O que a camada faz com os dados de entrada é armazenado nos pesos da própria camada, que são essencialmente uma série de números. Em termos técnicos, a transformação implementada por uma camada é parametrizada pelos seus pesos (os pesos são também apelidados de parâmetros de uma camada). Neste contexto, obter os parâmetros significa encontrar um conjunto de valores para os pesos de todas as camadas de uma rede, de forma que a rede mapeie corretamente as entradas de exemplo para os resultados esperados. O grande problema prende-se no facto de uma rede neuronal profunda poder conter dezenas de milhões de parâmetros, encontrar o valor correto para todos eles pode parecer uma tarefa complicada, já que a modificação do valor de um parâmetro afetará o comportamento de todos os outros.

Algo que ainda está por responder neste campo é qual o número ótimo de camadas para que o modelo tenha o melhor desempenho, já que o número está dependente do propósito. No caso de poucas camadas, o modelo terá menos sub-regiões para agrupar os pontos, de modo que muitos serão erradamente agrupados. Por outro lado, se o número de camadas for grande, há um grande risco de *overfitting*, o que irá causar uma diminuição na *performance*. Desta forma, a rede deverá ser alimentada com dados

suficientes para que o número de partições dos dados sejam suficientes e as camadas ocultas consigam, corretamente, separar os dados [9].

2.2.1 Redes neuronais (*Neural Networks*)

O cérebro humano é um exemplo de uma rede neuronal biológica capaz de realizar tarefas de perceção e controlo de maneira eficiente. Isso é possível graças à sua estrutura computacional altamente paralela e à sua capacidade de processamento de informação imprecisa. O cérebro é composto por milhões de neurónios interligados, onde cada um utiliza reações bioquímicas para receber, processar e transmitir informação.

Redes neuronais são sistemas inspirados por redes neuronais biológicas, isto é, aprendem a realizar funções ao serem apresentados a diversos conjuntos de dados sem qualquer tipo de norma para a realização da tarefa. Como referido, estas são capazes de aprender e, para isso, precisam de ser treinadas através de aprendizagem supervisionada (2.1.1), aprendizagem não supervisionada (2.1.2) ou aprendizagem de reforço (2.1.3). No entanto, é importante mencionar que para modelos de reconhecimento de imagem geralmente a estratégia utilizada é a aprendizagem supervisionada, [10].

2.2.1.1 Redes neuronais artificiais

As redes neuronais artificiais (Artificial Neural Network (ANN)) foram desenvolvidas como generalizações de modelos matemáticos de sistemas nervosos biológicos. O início do interesse nas redes neuronais surgiu após a introdução de neurónios simplificados por *McCulloch* e *Pitts* (1943), [3]. As unidades de processamento básicos das redes neuronais são chamados de neurónios artificiais ou nós. Num modelo matemático simplificado do neurónio, os efeitos das sinapses são representados por pesos que modulam o efeito dos sinais de entrada associados. A capacidade de aprendizagem de um neurónio artificial é alcançada ajustando os pesos de acordo com o algoritmo de aprendizagem escolhido.

O funcionamento de uma ANN está representada na Figura 2. Insere-se os dados de entrada, com estrutura de um vetor multidimensional, a camada de entrada irá distribuir esse vetor pelas camadas ocultas. Nesta fase, as camadas ocultas tomam as decisões tendo em conta a camada anterior e pesam a forma como uma escolha influencia o resultado final, dando-se então a aprendizagem. Ter várias camadas ocultas empilhadas é o que se denomina por aprendizagem profunda (DL) [10].

2.2.1.2 Redes neuronais convolucionais (CNN)

As redes neuronais convolucionais (Convolutional Neural Network (CNN)) são um tipo específico de rede neuronal que é projetada para processar dados que possuem uma estrutura em grade, como uma imagem. Estas são constituídas por quatro áreas chave para o seu funcionamento (ver Figura 7).

A camada de entrada que recebe os valores de píxeis brutos de uma imagem e as restantes camadas aplicam várias transformações a esses valores para extrair características que podem ser usadas para

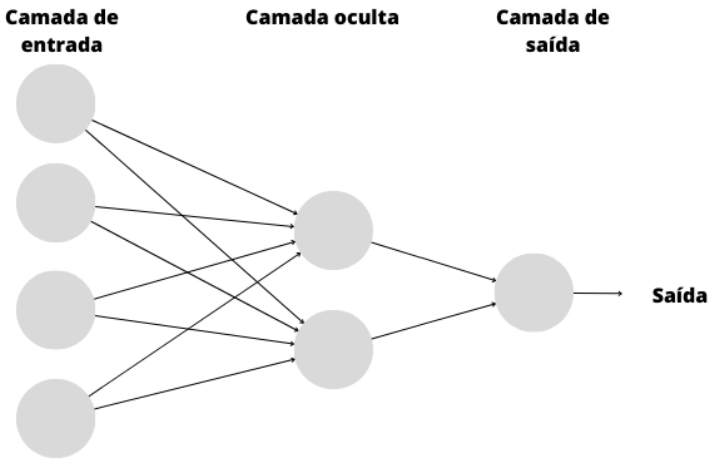


Figura 6: Rede neuronal Artificial

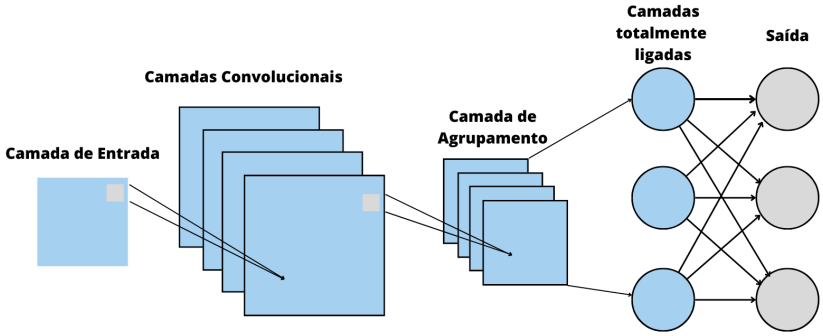


Figura 7: Rede neuronal Convolucional

classificação ou regressão. De seguida, as camadas convolucionais (estas são as principais unidades de construção de uma CNN) aplicam uma série de filtros nos dados de entrada para extrair características. Os filtros são pequenos, geralmente 3x3 ou 5x5 e deslizam sobre os dados de entrada, realizando um produto interno entre os pesos do filtro e os valores dos dados de entrada em cada posição. Isso resulta num conjunto de ativações, que então são transformadas por uma função de ativação.

As camadas de agrupamento são usadas para reduzir o tamanho espacial das ativações, resultando numa diminuição do número de parâmetros no modelo o que o torna computacionalmente mais eficiente. O agrupamento é geralmente realizado tomando o valor máximo ou médio de uma pequena janela das ativações.

Por fim, as camadas totalmente ligadas são semelhantes às de uma rede neuronal padrão e são usadas para produzir a saída final da CNN, que pode ser usada para classificação ou regressão.

2.3 Mineração de texto

Nos últimos anos a quantidade de informação digital tem aumentado, sendo que a maioria desta se encontra em textos não estruturados [11]. Ao contrário de textos estruturados, os computadores não conseguem simplesmente processar os textos e retirar informação, são necessários diversos métodos e algoritmos extremamente precisos para tratar deste tipo de informação. Desta forma o *text mining*, ou mineração de texto, tem vindo a ganhar popularidade, já que há uma crescente necessidade de transformar estes dados em conhecimento[12].

O *text mining*, é uma combinação de técnicas de processamento de linguagem natural (Natural Language Processing (NLP)), mineração de dados e ML, que tem a capacidade de extrair informações relevantes de documentos de texto. Uma das principais vantagens dessa abordagem é a capacidade de lidar com grandes volumes de texto não estruturado, permitindo analisar desde artigos científicos até simples publicações em redes sociais. Esta versatilidade e capacidade de retirar informações a partir de dados não estruturados ou semi-estruturados, torna possível uma melhor organização e uma mais eficiente aplicação dessas informações [13].

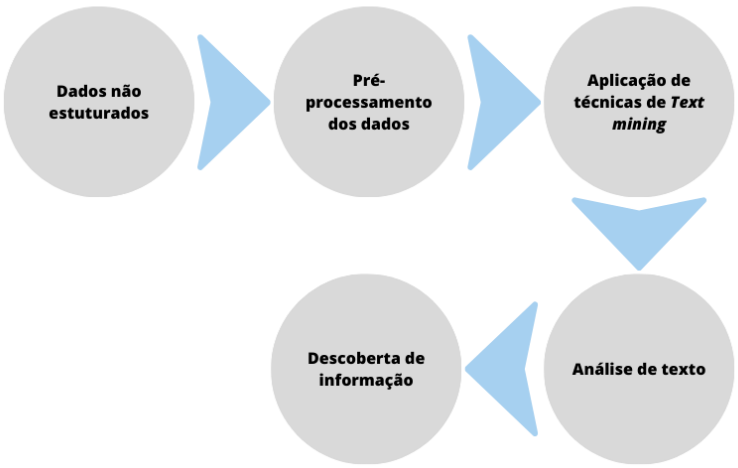


Figura 8: Esquema mineração de texto

Existem diversas técnicas e métodos para "ensinar" computadores a analisar, perceber e devolver o conhecimento de um texto, porém todas se baseiam no diagrama apresentado da Figura 8. Cada uma destas técnicas tem a sua utilidade, dependendo do problema em questão.

- **Extração de informação** - O propósito deste método é identificar e extrair entidades, eventos e relações. Ao contrário de nomes e lugares, que podem ser extraídos sem a compreensão do texto, este método foca-se na extração de informação presente na semântica do texto [11].
- **Categorização** - É um método de aprendizagem supervisionada que se baseia em textos previamente classificados numa ou mais classes e classifica novos textos [12]. O objetivo é treinar o

modelo com base em exemplos conhecidos e aplicá-lo a exemplos desconhecidos.

- ***Clustering*** - É geralmente utilizado para encontrar grupos de documentos com conteúdo similar. Como resultado obtém-se grupos de documentos. Apesar de parecido com a categorização, este método difere uma vez que classifica estes documentos sem a existência de grupos pré-definidos [12].

- **Visualização** - Este método utiliza uma escala de cores para identificar temas que um grupo de documentos, ou partes de um só documento. A visualização coloca grandes quantidades de texto numa hierarquia visual. Adicionalmente, o utilizador pode interagir com as cores, procurando por exemplo documentos que apenas tratem de desporto ou política.

- **Sumarização** - é uma técnica utilizada quando o problema se trata de reduzir o tamanho de um texto sem a perda de informação. Envolve diversas técnicas, como redes neuronais já referidas anteriormente, mas tal como estas a criação de um modelo é extremamente variável e dependente do tipo de dados que se está a analisar, neste caso específico, do tipo de texto que será sumarizado [11].

2.3.1 Pré-processamento de texto

Modelos de ML atuam inteiramente com vetores de números e por isso, para modelos de *text mining*, os textos precisam de ser convertidos em vetores através de um processo chamado vetorização. Para que este processo seja o mais eficiente possível, aplicam-se técnicas de pré-processamento de texto, tais como:

- *Tokenization* - Processo pelo qual se "parte" o texto em palavras, denominados *tokens*. Esta divisão é efetuada tento em conta espaços em branco ou pontuação entre as palavras [14]. Por exemplo, a frase "Olá, sou o João", depois da *tokenization*, resultaria em ["Olá", ",", "sou","o","João"]

- *Stop-words* - Técnica aplicada para reduzir a dimensionalidade do texto, removendo as palavras sem qualquer valor para o modelo. Palavras como pronomes e preposições são vistas como *stop-words* [15] Tendo como exemplo a mesma frase "Olá sou o João" depois da remoção das *stop-words*, o resultado seria seria "Olá João".

- Conversão para minúsculas - Mesmo assumindo que não haja diferença entre letras minúsculas e maiúsculas, o texto é todo convertido em minúsculas para homogeneizar a informação

- Redução de palavras à sua raiz - É a redução das palavras à sua raiz/radical. O propósito deste método é a remoção de vários sufixos para obter raízes/radicais correspondentes, reduzindo assim a capacidade computacional necessária [15]. Um exemplo prático, tendo como base a mesma frase "Olá sou o João" depois de aplicado o *stemming*, resulta em "Ola sou o Joa"

2.3.2 *Word embeddings*

Word Embedding é uma técnica na qual cada palavra ou frase é mapeada para um vetor de números reais de dimensão *N* [16]. Existem diversos métodos entre os quais o *Word2Vec* e o Global Vectors for Word Representation (GloVe), que têm como objetivo de traduzir palavras para inputs percetiveis para um modelo de ML.

2.3.2.1 *Word2Vec*

O *Word2Vec* é um algoritmo popular que mapeia palavras em vetores de valores reais, também conhecidos como *embeddings*. Esses *embeddings* capturam a semântica e a relação contextual entre as palavras. Existem duas principais abordagens quando se utiliza o *Word2Vec*: Continuous bag-of-words Network (Cbow) e *Skip-gram*.

A abordagem utilizando o modelo Cbow tenta prever a palavra que falta no meio de uma frase, por exemplo a frase "O cão comeu um osso", o modelo iria tentar prever a palavra "comeu". Este método é uma ferramenta bastante útil quando o objetivo é descobrir relações e similaridades entre palavras [16].

No caso do modelo *Skip-gram* apesar de apresentar uma arquitetura semelhante ao Cbow, ao invés de prever a palavra com base no contexto, tenta maximizar o contexto com base numa palavra. Utilizando a mesma frase, "O cão comeu um osso"a entrada no modelo seria apenas uma palavra da frase e o modelo tenta prever o resto da frase, repetindo este processo para todas as palavras da frase.

2.3.2.2 *Global Vectors for Word Representation*

Outra ferramenta poderosa de *embeddings* é o GloVe onde a principal ideia é tentar perceber a relação de co-ocorrência de palavras num enorme conjunto de textos ou documentos. O modelo pré-treinado utilizado em diversos projetos é treinado num vocabulário de 400.000 palavras composto pelo *Wikipedia* 2014 e o *Gigaword 5* e 50 dimensões para a apresentação das palavras [16]. O GloVe fornece também outros modelos pré-treinados para vetorização de palavras que dispõe de 100, 200 ou 300 dimensões e são treinados com ainda mais informação que o anterior, incluindo conteúdo presente no *Twitter*.

2.3.3 Modelos de classificação de texto

Devido à extensa quantidade de informação em documentos digitais, a classificação automática de textos é extremamente útil e já se tornou uma necessidade. Existem dois tipos de classificação de texto: uma primeira abordagem onde o objetivo passa por classificar um documento com base nos tópicos que este aborda, isto é, se um documento é referente a desporto, política, etc; outra abordagem é classificação por género, se é um artigo científico, notícia, análise crítica a um filme, entre outros [17]. O propósito independentemente da abordagem escolhida, é encontrar um classificador que se assemelhe ao máximo a um humano na tarefa de classificar documentos.

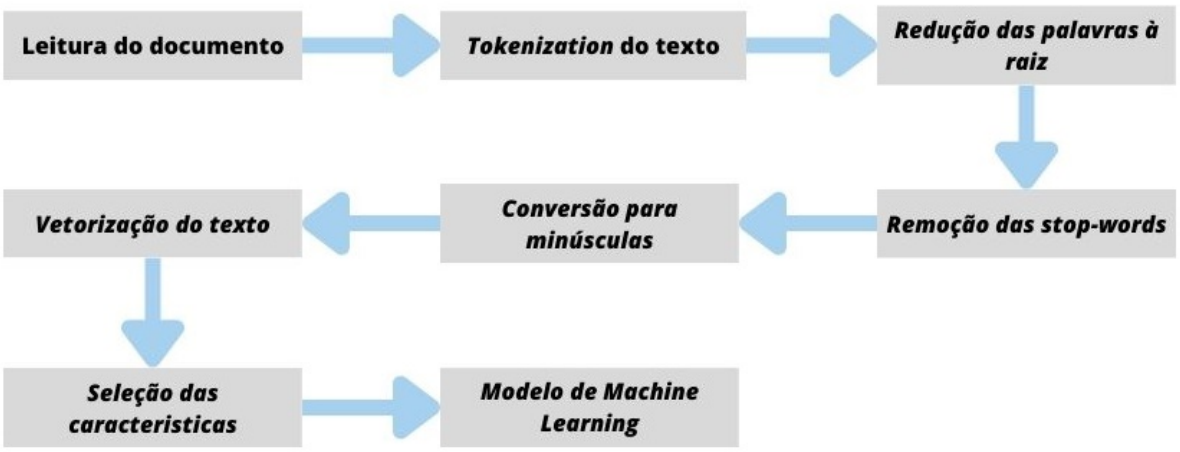


Figura 9: Processo de aplicação de um modelo de classificação de texto

2.3.4 Arquiteturas

A classificação de texto é uma tarefa fundamental no que diz respeito a NLP, que se baseia na atribuição de rótulos ou categorias a textos com base no seu conteúdo.

Com o aparecimento de arquiteturas pré-treinadas, como por exemplo o famoso modelo criado pela *OpenAI*, o Generative Pre-trained transformer (GPT), a tarefa de classificação de texto tornou-se mais acessível e eficaz. Estas arquiteturas são treinadas em conjuntos de texto não rotulados , o que facilita a captura de semântica e relação entre palavras. Posteriormente estes modelos podem ser ajustados, o chamado *finetuning* para tarefas especificas com conjuntos menores de texto rotulado. Esta capacidade de transferir o conhecimento obtido em textos genéricos para conjuntos mais pequenos e específicos representa uma das vantagens mais significativas das arquiteturas pré-treinadas para classificação de texto.

Serão agora apresentadas algumas arquiteturas conhecidas dentro da área de classificação de texto.

2.3.4.1 *Bidirectional Encoder Representations from Transformers*

Bidirectional Encoder Representations from Transformers (BERT) É um modelo de classificação de texto pré-treinado desenvolvido pela Google que Utiliza uma arquitetura do *Transformer*, que é baseada em redes neuronais profundas. Uma das principais características deste modelo é a capacidade de processar palavras num contexto bidirecional, isto é, considera tanto as palavras que aparecem antes como depois de uma determinada palavra numa frase. Isto ajuda na captura de significado nas palavras, impulsionando o seu desempenho em tarefas como classificação de texto, previsão da próxima palavra, etc.

Existem atualmente dois modelos pré-treinados BERT, o *BERTBASE* e o *BERTLARGE*. O *BERTBASE* é constituído por 12 camadas e contém um total de 110 milhões de parâmetros enquanto que o *BERTLARGE* tem o dobro das camadas e é composto por 340 milhões de parâmetros [18].

Com este modelo pré-treinado, adiciona-se uma nova camada não treinada em cima da camada de saída desta forma a tarefa de *fine-tuning* torna-se bastante mais rápida. O que acontece é que todas as camadas do modelo já estão pré-treinadas com uma quantidade enorme de dados e esta ultima adicionada irá ser treinada com base no conjunto de dados menor específico para o problema.

2.3.4.2 *Generative Pre-trained transformers*

GPT remete para um conjunto de modelos de linguagem pré-treinados desenvolvidos pela *OpenAI*. São modelos que foram treinados com uma base de dados enorme com informação de texto e podem ser aplicados em diversos problemas, incluindo classificação de texto. Uma vantagem em relação ao BERT, é a sua capacidade de geração de palavras, isto é, os GPT's são treinados igualmente para a previsão da próxima palavra com base em todas as anteriores.

O *GPT-2*, uma das versões deste modelo, é pré-treinado com milhões de textos retirados da internet e para a aplicação num problema específico é afinado com uma base de dados menor e com informação sobre o problema em estudo, como é possível ver na Figura 10.

Figura 10: Processo de treino do modelo *GPT2*

O *GPT-3*, outra versão do modelo, é treinado com uma mistura de bases de dados que totalizam 400 biliões de *tokens* e tem um máximo de 175 biliões de parâmetros [19]. A grande diferença face ao *GPT-2* é o facto deste modelo responder a comandos, isto é, é possível fazer "pedidos" ao modelo. O processo está descrito visualmente na Figura 11.

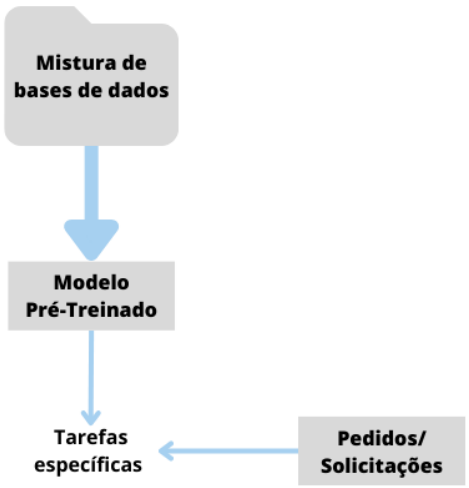


Figura 11: Processo de treino do modelo *GPT3*

2.4 Classificação de Imagem

A classificação de imagem é um campo do AI que trata de criar algoritmos para identificar objetos, padrões, pessoas, etc. Isto é feito através de técnicas de ML, como por exemplo as redes neuronais (2.2.1) que são usadas para treinar modelos e reconhecer padrões específicos nas imagens. Este campo do AIé usado em diversos ramos, como segurança, robótica e retalho. Devido à sua peculiaridade e diversidade de produtos, torna a tarefa de reconhecimento de objetos na área do retalho complexa. De forma a generalizar os desafios relacionados com a tarefa referida, estes podem ser divididos em quatro aspetos chave:

- Classificação em grande escala - O número de produtos diferentes que têm de ser identificados num supermercado são imensos e estão distribuídos por centenas de classes e de momento, os modelos já desenvolvidos apenas estão treinados para um número de classes que fica aquém do que é necessário. O *dataset* com melhor rácio entre número de classes e número de imagens é apelidado de *MS COCO* e contém fotos de oitenta classes de objetos [20], e cerca de trezentas mil imagens.

- Limitação de informação - Como já referido, abordagens que se baseiam em DL precisam de uma quantidade de dados enorme para o processo de treino. A recolha e rotulação de todos os produtos de um supermercado requer trabalho manual o que indica que a criação de uma base de dados com dados suficientes para treino é um trabalho que requer o seu tempo. Adicionalmente, as bases de dados existentes atualmente carecem de classes contempladas ou de número de imagens por objeto. Desta forma é possível concluir que um dos problemas atualmente é escassez de dados.

- Variação intraclasse - Identificação de subcategoria, é uma tarefa desafiante quando se tem em mente que: alguns produtos são extremamente semelhantes sendo a única diferença, por exemplo, um pequeno selo na embalagem, alguns produtos podem ter formas e aparências diferentes e fatores externos como por exemplo a iluminação [20]. Para esta distinção é preciso uma classificação cuidada de todos os produtos e sem isso, torna o uso de abordagens de DL muito mais exigentes computacionalmente.

- Flexibilidade - Dia após dia aparecem novos produtos nos supermercados de forma a atrair clientes e além disso, as aparências dos produtos já existentes vão mudando ao longo do tempo. Posto isto, o modelo de reconhecimento deve ser flexível de modo a não diminui *performance* a identificar antigos produtos quando lhe é fornecido um novo.

2.4.1 Arquiteturas

Existem diferentes tipos de arquiteturas de Redes Neuronais Convolucionais. Aqui são apresentadas algumas arquiteturas populares.

2.4.1.1 *You Only Look Once* (YOLO)

You Only Look Once (YOLO) é um algoritmo de deteção e classificação de imagens que foi apresentado em 2015 [21]. Dentro dos algoritmos de deteção de imagem o YOLO é conhecido pela sua rapidez uma vez que este apenas "vê" a imagem uma vez, daí o nome *You Only Look Once*. Este divide a imagem numa grade de células (SxS) e cada uma destas é responsável por detetar objetos dentro da sua área. Cada célula é uma classificador que usa uma rede neuronal para determinar se há algum objeto e se sim, onde é que este se encontra dentro da sua área. Atualmente há diversas versões do YOLO de forma a otimizar os resultados e a velocidade. Este algoritmo, apesar de bastante eficaz e veloz tem algumas restrições a nível espacial, isto é, cada célula da grade apenas pode prever dois objetos e apenas uma classe, o que limita o número de objetos que o modelo pode prever [22]. Desde a sua criação que o YOLO já conta com diversas versões, como a *YOLOv2*, *YOLOv3* e *YOLOv4*. No anexo I.1 está presente a arquitetura referente ao *YOLOv3 tiny*.

2.4.1.2 SSD (*Single Shot MultiBox Detector*)

Outro algoritmo bastante utilizado é o Single Shot MultiBox Detector (SSD), apesar de similar ao YOLO em termos de funcionamento. Foi projetado para ser mais preciso, principalmente em casos onde se trata de detetar objetos de formas diferentes. Da mesma forma que o YOLO, divide a imagem numa grade de células, porém ao invés de utilizar cada célula como um classificador, este contém mais camadas convolucionais com células de diferentes tamanhos para que facilite a tarefa de identificar objetos com tamanho e formas diferentes [23].

2.4.1.3 Visual Geometry Group - 16 (VGG-16)

O Visual Geometry Group (VGG)-16 é um modelo de redes neuronais convolucionais (CNN) que foi proposto em 2014 por *Zisserman* e *Simonyan*, da Universidade de Oxford [24]. É composto por 16 camadas convolucionais e camadas totalmente ligadas e é utilizado em tarefas de classificação de imagens. O modelo foi previamente treinado com um grande conjunto de dados de imagens chamado *ImageNet*, que contém milhões de imagens e milhares de classes, o que o torna bastante versátil e viável. Tem uma arquitetura bem definida, de fácil compreensão e é capaz de extrair características complexas de imagens com alta precisão. É um modelo bastante popular no campo da visão por computadores.

No entanto assim como outros modelos de CNN, o VGG-16 pode ser lento e exigir muito poder computacional para ser treinado em grandes conjuntos de dados. Este pode sofrer de *overfitting* se não for cuidadosamente ajustado.

2.4.1.4 EfficientNet

O *EfficientNet* é uma família de arquiteturas de redes neuronais convolucionais que tem como objetivo atingir o equilíbrio entre precisão e eficiência computacional. Este utiliza o método do coeficiente composto, utilizado para aumentar a escala dos modelos de forma mais simples e eficiente. Ao invés de aumentar arbitrariamente a profundidade, a largura ou a resolução, esta técnica avalia igualmente cada dimensão baseando-se num conjunto fixo de coeficientes [25]. A escala de largura (*width scaling*) está relacionada com o número de mapas de características nas camadas convolucionais. A escala de profundidade (*depth scaling*) é referente ao número de camadas na rede, enquanto a escala de resolução (*resolution scaling*) afeta o tamanho da entrada da imagem. Esta arquitetura já conta com sete versões, sendo que a última, *EfficientNet-B7*, superou os modelos convencionais na base de dados *ImageNet*. Atingiu uma precisão de 84.3% enquanto que, em comparação, é 8.4 vezes menor e 6.1 vezes mais rápido que o melhor modelo *ConvNet* existente [26].

2.4.1.5 Inception

Inception é uma arquitetura de uma rede neuronal convolucional profunda que foi vencedora do desafio "*ImageNet Large-Scale Visual Recognition Challenge*" (ILSVRC14) e foi desenvolvida, praticamente na sua maioria por investigadores da *Google*. É bastante preciso, eficaz e bastante mais leve computacionalmente em comparação com por exemplo, o VGG16 [27].

A principal inovação do *Inception* V3 é a introdução de módulos de convolução denominados "*Inception modules*". Esses módulos foram projetados para capturar informações em diferentes escalas e resoluções, permitindo que a rede identifique características complexas numa imagem. Os módulos *Inception* utilizam convoluções com diferentes tamanhos de filtros (1x1, 3x3, 5x5) e *pooling* em paralelo, combinando as saídas num único tensor (um tensor é uma estrutura de dados multidimensional que

organiza informações de forma eficiente, representando dados em várias dimensões). Isso ajuda a extrair informações de forma mais eficiente e melhora a capacidade da rede para aprender representações mais ricas.

2.5 Optical Character Recognition (OCR)

O Reconhecimento Óptico de Caracteres, mais conhecido como Optical Character Recognition (OCR) é uma tecnologia que permite a extração de informação de documentos físicos ou digitalizados, tornando-a editável, se necessário. É um campo vasto com imensa aplicabilidade como por exemplo em bancos para processar cheques sem o envolvimento humano e na indústria da saúde para transferir informação de pacientes em papel para uma base de dados eletrónica, é também utilizado na ferramenta Completely Automated Public Turing Test To Tell Computers and Humans Apart (CAPTCHA) [28], e no *Google Lens*.

Na Figura 12 está presente um esquema de possíveis utilizações para o OCR onde este analisa diferentes tipos de ficheiros com texto no seu conteúdo e retorna um outro ficheiro de texto editável.

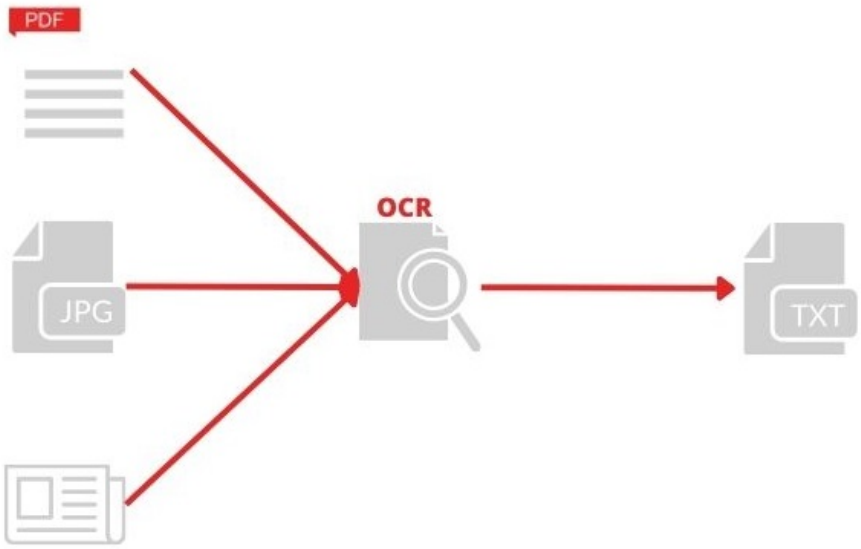


Figura 12: Esquema de utilização de OCR

Capítulo 3

Métodos e metodologias

Este capítulo explica todo o *software* utilizado no desenvolvimento da dissertação, bem como os métodos utilizados para o tratamento de dados e criação dos modelos.

Para que a leitura não se torne extensiva e repetitiva, dado que a implementação do código é extremamente semelhante em cada categoria, apenas será apresentada a implementação para a categoria de Vegetariano.

3.1 Ferramentas Utilizadas para Desenvolvimento

Para que o projeto se tornasse mais completo, foram utilizadas diferentes abordagens para o mesmo, estas são a classificação de imagem, classificação de texto e, por fim, um *ensemble* das duas anteriores. Nas secções abaixo estas abordagens serão explicadas.

3.1.1 *Microsoft Azure*

O *Microsoft Azure* é uma plataforma na nuvem que oferece uma diversa gama de ferramentas para construir, implementar e realizar a manutenção de aplicações e serviços. É atualmente uma das principais plataformas do mercado e é utilizada por várias empresas no mundo todo.

Este oferece tanto suporte para uma ampla variedade de linguagens de programação, como por exemplo *python*, R, *Java*, entre outros, bem como serviços de segurança, como deteção de ameaças e proteção contra Distributed Denial of Service (DDoS) de forma a manter todos os dados e informações seguras.

3.1.2 Azure Databricks

Azure Databricks é um conjunto de ferramentas unificadas usadas para processar, analisar e modelar bases de dados com soluções desde Business Intelligence (BI) a ML. Esta ferramenta providencia uma interface com instrumentos para a maioria das tarefas com dados, das quais:

- Programação e gestão de fluxos de trabalho de processamento de dados
- Trabalhar em Structured Query Language (SQL)
- Gerar *dashboards* e visualizações
- Ingestão de dados
- Gestão da segurança
- Exploração de dados
- Modelação ML
- Versionamento de código com o *Github*

Com esta informação, entende-se que as utilizações de *Azure Databricks* são bastante diversificadas e podem ser aplicadas em diversos casos de usos diferentes.

3.1.3 Spark

O *Apache Spark* é um motor *open-source* de processamento e consultas distribuídas proporciona flexibilidade e extensibilidade de *MapReduce* (modelo de processamento desenhado para processar grandes volumes de dados) mas com uma grande velocidade.

Permite ao utilizador ler, transformar, agregar dados e possibilita também o treino e implementação de modelos estatísticos. Está acessível em diversas línguas de programação, como *Java*, *Python*, *R* e *SQL* e é bastante usado na construção de aplicações ou bibliotecas, na implementação das mesmas em *clusters* ou para analisar dados através de *notebooks* (*Databricks notebooks*) [29].

Na Figura 13 é possível ver o esquema de como o *spark* funciona. Há um serviço, isto é, um pedido por parte do utilizador, com isto o *driver* ao qual está a ser feito o pedido divide essa tarefa por trabalhadores que realizam tarefas pequenas, tornando o processo mais rápido e menos pesado computacionalmente.

3.1.4 Python

Python é uma linguagem de programação de alto nível, conhecida pela sua sintaxe fácil e legível. A sua criação iniciou-se nos anos 90 por *Guido Van Rossum* como alternativa à linguagem de programação

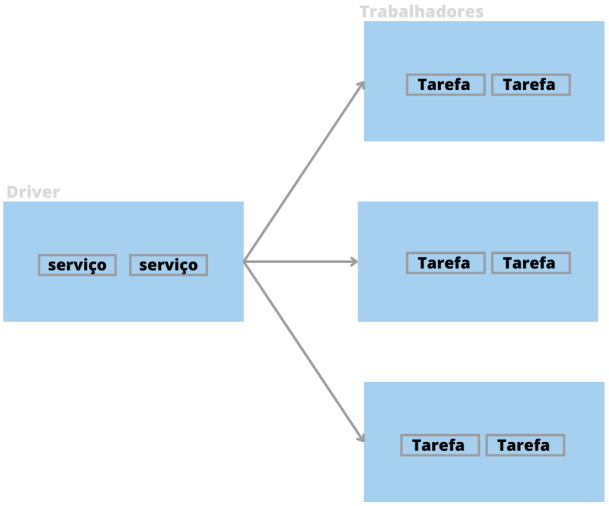


Figura 13: Esquema de funcionamento *spark*

ABC [30].

Possui estruturas de dados de alto nível que combinadas com o seu *dynamic tipyng* e a sua ligação dinâmica (o tipo das variáveis não necessita de ser definido pelo utilizador) demonstram a sua sintaxe intuitiva, o que torna esta linguagem cativante para o desenvolvimento de aplicações e *scripting* [31].

No projeto esta linguagem foi utilizada para o desenvolvimento de todo o código necessário para a criação e treino dos modelos de imagem, texto e o *ensemble*.

3.1.5 SQL

SQL (*Structured Query Language*) é a forma de comunicar com uma base de dados [32]. É empregada geralmente quando há uma necessidade de adicionar ou "refrescar" a informação ou então para adquirir alguma informação concreta de uma base de dados.

Neste projeto o SQL foi utilizado para obter os dados necessários, como vai ser explicado posteriormente.

3.2 Classificação de Imagem

Uma das abordagens utilizadas neste projeto foi a classificação de imagem com base em algoritmos mencionados no capítulo 2.4. Para tal foram necessários diversos procedimentos e aplicadas várias técnicas listadas e explicadas abaixo.

Para qualquer modelo de ML os elementos mais importantes e indispensáveis são os dados, num modelo de classificação de imagem não é diferente. Numa primeira instância, o objetivo foi perceber onde existiriam dados úteis para esta abordagem, isto é, imagens de produtos do Continente. Foram então

realizadas reuniões com diferentes equipas dentro da MC para tentar angariar o máximo de informação para a progressão do projeto.

Para simplificar a explicação neste capitulo, a informação que será apresentada será referente ao problema de classificação na classe de Vegetariano.

3.2.1 Dados

A aquisição e tratamento dos dados são indispensáveis para o sucesso de um modelo e nas secções que se seguem estão apresentados todos os processos para a recolha e tratamento da informação utilizada nesta abordagem.

3.2.1.1 Aquisição das Imagens

No decorrer das reuniões foi comunicado a existência de uma base de dados que continha as imagens de grande parte dos produtos vendidos no Continente.

Antes da obtenção das imagens foi criado uma *query* de SQL para guardar a lista de códigos de produtos (Stock Keeping Unit (SKU)) das categorias de Peixaria e Talho, Alimentar e Frutas&Legumes, Charcutaria&Queijos e *PAD TAWAY*, vendidos entre o dia 1 de janeiro de 2021 e 1 de abril de 2023 nas insígnias Continente Bom Dia, Continente Modelo, Continente e Continente *On-line*.

Figura 14: Imagem presente no banco de dados

Numa análise manual destas fotos, notou-se que alguns produtos não tinham imagem disponível, como é possível ver abaixo, o que seria um contratempo na extração das imagens.

Foi desenvolvido um *script* de *python* que automaticamente acede ao banco de dados referido e extrai as imagens dos produtos presentes na lista acima mencionada para uma pasta no *container* do *Azure*. Como certos produtos não continham nenhuma imagem, foi acrescentada ao código uma exceção para que sempre que aparecesse a imagem presente da Figura 15, esta não ser extraída. As imagens extraídas continham todas o formato *Product_sku.jpg*

Figura 15: Produto sem imagem disponível

3.2.1.2 Classificação por categoria

Obteve-se também uma base de dados *excel* com grande parte dos produtos do Continente. Alguns grupo deste produtos continha uma classificação de 0 ou 1 nas categorias de biológico, sem glúten, sem lactose, sem açúcar, *vegan* e vegetariano, onde 0 representa não "pertencer" à categoria e 1 representa "pertencer" à categoria. O número de produtos com classificação encontra-se na Tabela 1.

	Pertence - 1	Não Pertence - 0
Biológico	1320	3837
Sem Glúten	5141	2842
Sem Lactose	5080	2972
Sem Açúcar	3494	3200
Vegan	4161	2994
Vegetariano	6566	2474

Tabela 1: Produtos com classificação nas categorias

3.2.1.3 Divisão dos dados em Treino, Teste e Validação

De forma a possuir dados para treinar, validar e testar os modelos, estes tiveram de ser divididos em três grupos distintos com base no código fornecido na listagem abaixo (3.1).

```
1 train_file_names, test_file_names = train_test_split(file_names, test_size
    =0.1, random_state=41)
2 train_file_names, val_file_names = train_test_split(train_file_names,
    test_size=0.22, random_state=41)
```

Listagem 3.1: Código utilizado na divisão dos dados em Treino/Teste/Validação

Os dados foram divididos tendo por base o principio de 70/20/10, isto é, 70% dos dados para treino, 20% dos dados para validação e 10% dos dados para teste. O trecho do código referente ao *random_state=41* serve apenas para garantir que se obtém sempre a mesma divisão dos dados.

Como é possível verificar na Tabela 2, os dados pertencentes ao problema de classificação de Vegetariano são desbalanceados, isto é, há um número diferente de observações em cada classe. Para tentar colmatar este problema, foi aplicado um *oversample* da classe minoritária.

	Pertence - 1	Não Pertence - 0
Treino	4744	1764
Validação	1319	438
Teste	647	257

Tabela 2: Observações por classe depois da divisão Treino, Teste e Validação

3.2.1.4 *Oversample* da classe minoritária

Na base de dados de treino foi detetado um desbalanceamento de classes e, para tratar esse problema foi utilizado o *oversample*, isto é, replicar imagens pertencentes à classe minoritária até ao número de imagens nas duas classes ser igual, neste caso, ser 4744.

3.2.1.5 *Data Augmentation* (Tratamento de Imagem)

De forma a generalizar melhor a aprendizagem do modelo, as imagens obtidas foram sujeitas a uma série de tratamentos (*Image Augmentation*). De forma a facilitar o entendimento das alterações efetuadas, foi usado como exemplo a Figura 16.

Esses tratamentos foram aplicados com base no código presente na Listagem 3.2.

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True,
    fill_mode='constant')
```

Listagem 3.2: Código relativo a *Data Augmentation*

3.2.1.5.1 Rotação

O primeiro tratamento a ser aplicado foi a rotação com parâmetro 10. Como o próprio nome indica a imagem é rodada num intervalo, no caso em concreto, de -10°a 10°, como se verifica na Figura 17.

Figura 16: Imagem sem alterações

Figura 17: Transformação Rotação

3.2.1.5.2 Translação

Na transformação de translação foram aplicados dois tipos de translação, horizontal e vertical, como é possível verificar nas Figuras 18 e 19, respetivamente. Para estas alterações foi utilizado valor do parâmetro foi 0.1, que significa que as imagens são transladadas até 10% da largura ou tamanho da imagem original, dependendo do tipo de translação que é aplicada.

Figura 18: Transformação Translação Horizontal

3.2.1.5.3 Distorção

Esta alteração refere-se à alteração da inclinação dos pixeis da imagem numa certa direção. O valor escolhido foi de 0.1, significando que os pixeis da imagem sofrem uma distorção de até 0.1°. Alguns exemplos desta técnica estão presentes na imagem 20.

3.2.1.5.4 *Zoom*

Esta alteração serve para ampliar ou reduzir a imagem em até uma certa percentagem desejada. No caso concreto, a percentagem escolhida foi de 10%. Algumas imagens com esta alteração estão presentes na Figura 21.

3.2.1.5.5 Inversão

A inversão, trata-se de inverter horizontalmente a imagem. Na Figura 22 está presente uma comparação entre a imagem original e a imagem invertida.

Figura 19: Transformação Translação Vertical

Figura 20: Transformação Distorção

Figura 21: Transformação Zoom

Figura 22: Transformação Inversão

3.2.2 Arquiteturas

Para o desenvolvimento do modelo, foram testadas duas arquiteturas diferentes sendo estas a *EfficientNetB0* e a *VGG16*. Estas foram implementadas de duas formas distintas, inicialmente para a tarefa de classificação, isto é, serem treinadas sobre os dados e posteriormente classificar os produtos. Numa segunda abordagem estas foram utilizadas como extratoras de características e alimentadas a um SVM, utilizado como classificador.

3.2.2.1 *EfficientNet*

A primeira arquitetura testada foi a *EfficientNet* e como já referido na secção 2.4.1.4, *EfficientNet* é uma família de arquiteturas que visa atingir um equilibro entre precisão e eficiência computacional. Para este projeto, foi escolhida a versão B0 uma vez que era a que apresentava uma melhor eficiência computacional.

Esta arquitetura foi testada usando diferentes hiperparâmetros tais como *learning rate* que controla o tamanho de cada "passo" durante o treino, taxa de *dropout* que desliga aleatoriamente neurónios de forma a evitar o *overfit* e *batch size* que determina o número de exemplos a utilizar em cada iteração do treino. Utilizou-se também diferentes otimizadores, nomeadamente o *Adam* e o Stochastic Gradient Descent (SGD). Inicialmente foi utilizada a arquitetura *EfficientNetB0* e "descongelaram-se" as camadas da mesma, para que se possa ajustar melhor aos dados, como é possível ver na Listagem 3.3.

```
1 base_model = EfficientNetB0(weights=pesos, include_top=False, input_shape=(
      img_width, img_height, 3), classes=2)
2
3 for layer in base_model.layers:
4         layer.trainable = True
```

Listagem 3.3: Excerto do código relativo ao carregamento do modelo

Posteriormente adicionaram-se algumas camadas densas com diferentes valores de neurónios associados, sendo estes 512, 256 e 128, valores que produziram melhores resultados considerando a complexidade do problema e dos dados, para que ajudasse o modelo a perceber padrões mais complexos. Foram adicionadas também camadas com taxas de *dropout* variáveis para evitar o *overfitting* do modelo. Também como medida para evitar o *overfit* foi adicionado um *callback*, sendo este o *earlystopping* que monitoriza a métrica de precisão em validação e com uma paciência de 10 épocas, como é possível verificar na Listagem 3.4.

```
1 early_stopping = EarlyStopping(monitor='val_accuracy', patience=10,
      restore_best_weights=True)
```

Listagem 3.4: Código relativo ao *earlystopping*

3.2.2.2 VGG-16

Como método de comparação, foi testada também a arquitetura VGG-16 (secção 2.4.1.3) para entender qual produzia melhores resultados. Para garantir que apenas as arquiteturas seriam diferentes, os testes efetuados com as duas arquiteturas foram iguais, isto é, mesmas taxas de *dropout*, *batch size*, *learning rate* e otimizadores.

```
1 base_model = VGG16(weights='imagenet', include_top=False, input_shape=(
      img_width, img_height, 3), classes=2)
```

Listagem 3.5: Código do código relativo carregamento da arquitetura VGG-16

3.2.3 SVM

Ambas as arquiteturas foram utilizadas não como classificadoras, mas sim como extratoras de características das imagens, sendo estas posteriormente usadas para alimentar um SVM para efetuar a classificação.

Este processo inicia-se com o *load* das arquiteturas, descrito nas listagens 3.5 e 3.3. É importante notar que o parâmetro *include_top=False* é estritamente necessário uma vez que este remove as camadas de classificação da arquitetura e uma vez que o objetivo é utilizá-las como extratoras de características, este parâmetro é imprescindível. De seguida as arquiteturas foram utilizadas para extraírem as características dos dados de treino, teste e validação, este passo está presente na Listagem 3.6.

```
1 train_features = model.predict(np.array(train_images))
2 test_features = model.predict(np.array(test_images))
3 val_features = model.predict(np.array(val_images))
```

Listagem 3.6: Excerto do código relativo à extração de caracteristicas

Geralmente os dados de entrada num SVM estão no formato de um vetor uni-dimensional, dessa forma, foi necessário reajustar o formato dos dados, reajuste esse explícito na Listagem 3.7. Com esta alteração os dados que formavam um *array* de matrizes passaram a formar um *array* de vetores uni-dimensionais, desta forma preparados para alimentar um SVM.

```
1 train_features = train_features.reshape(train_features.shape[0], -1)
2 test_features = test_features.reshape(test_features.shape[0], -1)
3 val_features=val_features.reshape(val_features.shape[0], -1)
```

Listagem 3.7: Código relativo à reformatação dos dados

3.3 Classificação de Texto

Para que o projeto se tornasse mais completo e viável foi utilizada uma outra estratégia, a Classificação de Texto. Para isto foram utilizadas diversas técnicas de NLP e CNN.

3.3.1 Dados

A qualidade e tratamento dos dados são essenciais para o êxito de um modelo e dessa forma, todos os processos que os dados foram submetidos encontram-se nas secções seguintes.

3.3.1.1 Descrições dos Produtos

Decidiu-se utilizar todas as descrições referentes ao produto, isto é, a descrição do próprio produto, da marca, da unidade de negócio, da sub-categoria, categoria e departamento, para isso, foi desenvolvido uma *query* em SQL que devolvia todas estas informações. Esta *query* assemelha-se à que foi utilizada na secção 3.2.1.1, com a diferença que ao invés de devolver apenas o SKU, devolve todas as descrições acima mencionadas.

3.3.1.2 Classificação por categoria

Para obter a informação das classificações por categoria utilizou-se a mesma base de dados referenciada na secção 3.2.1.2. Esta informação é indispensável ao modelo para que seja possível realizar o treino. A divisão das classificações por categoria encontra-se presente na tabela 1.

3.3.1.3 Pré-processamento

Em análise de texto e NLP a qualidade dos dados de entrada tem um papel crucial no sucesso de qualquer tarefa. Antes de alimentar os dados ao modelo para que este aprenda a classificar, é necessários preparar esses mesmos dados. Esta secção explora todo o tratamento da informação.

A biblioteca utilizada foi a biblioteca *re* [33], esta é bastante conhecida no que toca a manipulação de *strings*, dessa forma, foi extremamente útil neste pré-processamento do texto. Este pré-processamento foi constituído por diversos passos que se basearam em remover todos os números, caracteres especiais, quantidades (gr,kg, l, etc.), espaços antes do texto e *stopwords* portuguesas, outro passo foi colocar todas as palavras em letra minúscula. O código que representa este tratamento está presente na Listagem 3.8.

Encontra-se abaixo, na Figura 23 e 24 respetivamente, um exemplo de uma observação antes do tratamento. Verifica-se que apenas toda a informação não necessária foi eliminada, mantendo apenas a informação importante do produto.

```
stopwords_pt = set(stopwords.words('portuguese'))

def preprocess_text(text):

    text = unidecode(text)

    text = re.sub(r'^\d+\s*-\s*', '', text)

    text = re.sub(r'\b\d+(\.\d+)?\s*(kg|g|ml|cap|gr|un|sq|saq|l|ca|lt?)\b',
        '', text, flags=re.IGNORECASE)

    text = re.sub(r'\b\d+\*?\s*[,.!?"\']?\s*$', '', text)

    text = re.sub(r'\b[a-zA-Z]+\d+\b', '', text)


    text = ' '.join(word for word in text.split() if word.lower() not in
       stopwords_pt)

    return text
```

Listagem 3.8: Código relativo ao processamento do texto

7622124| 10 - ALIMENTAR| 1403 - BEBIDAS QUENTES |140304 - CÁPSULAS |14030404 - COMP. NESPRESSO ALU.| 7622124 - CAFÉ LAVAZZA RISTRETTO 10 CAP |LAVAZZA

Figura 23: Observação pré-processamento

alimentar bebidas quentes capsulas comp. nespresso alu. cafe lavazza ristretto lavazza

Figura 24: Observação pós-processamento

3.3.1.4 Treino, Teste e Validação

Tal como na classificação de imagem os dados foram divididos em três grupos, nomeadamente Treino, Teste e Validação. Os dados foram igualmente divididos em 70% para treino, 20% para validação e 10% para teste. Apesar de os dados serem provenientes de bases de dados diferentes, foi possível constatar que o problema de classes desbalanceadas existia novamente, como é possível ver na Figura 4.

	Pertence - 1	Não Pertence - 0
Treino	4719	1699
Validação	1342	501
Teste	505	274

Tabela 3: Observações por classe depois da divisão Treino, Teste e Validação

3.3.1.5 *Oversample* da classe minoritária

Para resolver o problema das classes desbalanceadas em treino utilizou-se a técnica conhecida na área de tratamento de classes desbalanceadas, o Synthetic Minority Over-sampling Technique (SMOTE). Esta técnica cria exemplos sintéticos da classe minoritária, de forma a equilibrar as classes.

Na Listagem 3.9 está visível o código utilizado para igualar as observações nas classes. Na primeira linha definiu-se a estratégia de amostragem, isto é, o número pretendido de observações em cada classe. Uma vez que a classe positiva é a classe com maior número de observações, será esse o valor pretendido. De seguida aplicou-se essa estratégia aos dados de treino e desta forma as classes já se encontraram balanceadas.

```
1 sampling_strategy = {0: np.bincount(train_labels)[1], 1: np.bincount(
    train_labels)[1]}
2
3 smote = SMOTE(sampling_strategy=sampling_strategy)
4
5 train_sequences_resampled, train_labels_resampled = smote.fit_resample(
    train_sequences, train_labels)
```

Listagem 3.9: Utilização do SMOTE para equilibrar as classes

3.3.2 *Embeddings*

Os *Word Embeddings* são técnicas utilizadas para mapear frases ou palavras para vetores de valores reais. O método escolhido foi o GloVe, mais em específico a versão *GloVe.6B.300D*, treinada com seis mil milhões de *tokens*. Esta escolha deveu-se não só à sua riqueza semântica, diversidade de relações entre palavras e resistência à ambiguidade das palavras, mas também porque era a versão mais "leve" computacionalmente, uma vez que as restantes versões tinham sido treinados em mais do triplo dos *tokens*.

O processo iniciou-se com a criação de um dicionário (*embedding_index*) para o armazenamento dos *embeddings* do GloVe. Este código percorre as linhas do arquivo pré-carregado GloVe e divide as linhas em palavras, juntamente com o seu vetor correspondente, posteriormente armazena essa informação no dicionário criado.

De seguida foi criada uma matriz de *embeddings* para ajudar o modelo a capturar a semântica de algumas palavras. Esta matriz é criada percorrendo todas as palavras presentes no *word_index* (lista previamente definida com todas as palavras do vocabulário presente na base de dados dos produtos), e caso exista uma correspondência no GloVe, esta será copiada para a respetiva posição da matriz de *embeddings*.

3.3.3 Arquiteturas

Para a classificação de texto foram testadas duas abordagens diferentes, uma com a utilização dos *embeddings* do GloVe e de uma CNN e outra utilizando a arquitetura pré-treinada BERT. Esta secção mostra a implementação de ambas as abordagens, bem como uma explicação das mesmas.

3.3.3.1 CNN

Na primeira abordagem foi utilizada uma CNN implementada manualmente, isto é, sem utilização de uma arquitetura pré-treinada. Esta CNN é composta por uma camada de entrada, formada pela matriz de *embeddings* anteriormente referida. Incluíram-se quatro camadas de convolução com filtros de diferentes tamanhos e a função de ativação Rectified Linear Unit (ReLU) de forma a tentar captar diferentes padrões no texto. Posteriormente adicionou-se uma camada de *Max Pooling* de forma a reduzir a dimensionalidade e a aumentar a eficiência computacional. Adicionaram-se também duas camadas densas, uma com 64 e outra com 128 neurónios com função de ativação ReLU e regularização L2 (ou ridge) com valor de 0.001. O objetivo destas camadas é "aprender" os padrões captados pelas camadas convolucionais anteriores. Por fim incorporou-se uma última camada densa, apenas com 1 neurónio e função ativação *sigmoid*, utilizada em problemas binários já que esta produz uma probabilidade desta de pertencer à classe positiva. O otimizador escolhido foi o *ADAM* com uma *learning_rate* de 0.00001. Como função de perda, a escolha foi a *binary_crossentropy*, uma vez que esta é apropriada para problemas binários. O código utilizado para a implementação desta CNN está presente na Listagem 3.10.

É importante mencionar que todos os valores utilizados de regularização, neurónios e *learning_rate* foram provenientes de diversos testes.

3.3.3.2 BERT

Outra abordagem foi através da utilização da arquitetura pré-treinada BERT. A versão empregue foi a *bert-base-portuguese-cased*. A decisão de utilizar a versão *base* ao invés da versão *large* baseou-se no peso computacional, dado que esta possui por volta de três vezes mais parâmetros que a versão *base*.

A implementação do modelo nesta abordagem encontra-se na Listagem 3.11. Este iniciou-se com o carregamento da arquitetura pré-treinada BERT, onde *PATH* representa a diretoria do *container* do *Azure* que contém a mesma. De seguida foram definidas as camadas do modelo treináveis para que este se possa adaptar melhor aos dados utilizados, juntamente com as camadas de entrada e saída do mesmo. Foi definida também uma camada com taxa de *dropout* com valor de 0.5 para que ajude a evitar o *overfit*. Em termos de camadas densas, foram adicionadas duas, com 64 e 128 neurónios, ambas com regularização L2 de valor 0.001 e função ativação ReLU. Dado que é um problema de classificação binária, incluiu-se outra camada densa com apenas um neurónio e função ativação *sigmoid* para que esta produza uma probabilidade. Utilizou-se como otimizador o *Adam* com um *learning_rate* de 0.00001 e a função *binary_crossentropy* que é usualmente utilizada em problemas de classificação binária.

```
model = Sequential()
model.add(Embedding(input_dim=num_words, output_dim=embedding_dim,
    input_length=max_sequence_length,
                    weights=[embedding_matrix], trainable=False))

num_filters = 256
filter_sizes = [1,3,5,7]
for filter_size in filter_sizes:
    model.add(Conv1D(filters=num_filters, kernel_size=filter_size,
        activation='relu'))

model.add(GlobalMaxPooling1D())
model.add(Dense(64, activation='relu',kernel_regularizer=tf.keras.
    regularizers.l2(0.001)))
model.add(Dense(128, activation='relu',kernel_regularizer=tf.keras.
    regularizers.l2(0.001)))
model.add(Dense(1, activation='sigmoid'))

optimizer = Adam(learning_rate=learning_rate)
model.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=['
    accuracy'])
```

Listagem 3.10: CNN utilizada na classificação de texto

```
bert_model = TFBertModel.from_pretrained("PATH/Bert")
bert_model.trainable = True

# Build the BERT-based model
input_ids = tf.keras.layers.Input(shape=(max_sequence_length,), dtype=tf.
    int32)
bert_output = bert_model(input_ids)[0]
dropout = tf.keras.layers.Dropout(0.5)(bert_output)
dense = tf.keras.layers.Dense(64, activation='relu', kernel_regularizer=tf.
    keras.regularizers.l2(0.001))(dropout)
dense = tf.keras.layers.Dense(128, activation='relu', kernel_regularizer=tf
    .keras.regularizers.l2(0.001))(dropout)

output = tf.keras.layers.Dense(1, activation='sigmoid')(dense)
model = tf.keras.Model(inputs=input_ids, outputs=output)

# Compile and train the model
optimizer = Adam(learning_rate=0.00001)
model.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=['
    accuracy'])
```

Listagem 3.11: Criação do modelo com base na arquitetura BERT

3.4 *Ensemble*

Uma terceira abordagem foi um *ensemble* das duas anteriores, isto é, a junção de um modelo de classificação de imagem e classificação de texto de forma a tentar obter os melhores resultados. Esta junção foi com base numa regressão logística que utiliza as previsões de ambos os modelos como dados de entrada.

3.4.1 Dados

Nesta abordagem os dados de entrada não foram diretamente obtidos numa base de dados, estes são compostos pelas previsões dos modelos de classificação de imagem e classificação de texto. Isto é, ao invés de se alimentar o modelo com as imagens e as descrições do produto, alimenta-se com as probabilidades obtidas com as previsões dos modelos das duas abordagens mencionadas anteriormente. Dessa forma todo o procedimento de tratamento de imagem e texto foi o mesmo que nas abordagens anteriores. Ainda assim, para garantir a coesão dos dados foi feito um *"merge"* das bases de dados referentes a cada uma das abordagens, isto é, junção das duas listas de SKU.

3.4.1.1 Treino e Teste

Dado que o treino (ou *fit*) da regressão logística apenas aceita como entrada os dados de treino, isto é, as probabilidades dos dois modelos e as respetivas classes (ou *labels*), os dados foram apenas divididos em dois grupos, onde 80% dos dados seriam para treino e 20% para teste.

	Pertence - 1	Não Pertence - 0
Treino	5368	1958
Teste	1342	501

Tabela 4: Observações por classe depois da divisão Treino e Teste

3.4.2 Implementação

A implementação da regressão logística iniciou-se com a previsão dos modelos de imagem e texto. Como o comando *"predict_proba"* retorna um *array* com duas colunas, foi necessário efetuar algumas transformações para que as probabilidades do modelo de imagem estivessem na forma desejada, como é possível ver na Listagem 3.12. Por fim para juntar a informação num único *array*, utilizou-se o comando *hstack* da biblioteca *numpy* para criar um *array* onde a primeira coluna contem as probabilidades do modelo de texto e a segunda as probabilidades do modelo de imagem.

Para a definição dos parâmetros a aplicar, foram efetuados diversos testes e os que produziram melhores resultados foram com a utilização de uma regularização L2 com valor 0.001 e o *solver saga*.

```
probabilidades_texto = modelo_texto.predict(train_text)
probabilidades_imagem = modelo_imagem.predict_proba(train_features)
probabilidades_imagem = probabilidades_imagem[:, 1]
probabilidades_imagem = probabilidades_imagem.reshape(-1, 1)

classes = train_labels_resampled

features = np.hstack((probabilidades_texto, probabilidades_imagem))

classificador = LogisticRegression(max_iter=300, penalty='l2', C=0.001,
    fit_intercept=True,solver='saga', verbose=1)
```

Listagem 3.12: Criação do modelo de *ensemble*

3.5 Extração de texto de uma imagem

Uma quarta abordagem testada foi a extração de texto de imagem, onde o objetivo passou por extrair da imagem do produto informação que pudesse complementar o modelo de classificação de texto.

Foi utilizada uma biblioteca *open source* chamada *easy OCR*. A escolha deveu-se não só ao facto de ser *open source*, mas também por suportar diferentes idiomas, incluindo o português.

A implementação deste algoritmo passou por definir o idioma pretendido para o OCR, carregar o mesmo com o auxilio do código *easyocr.Reader* da biblioteca e, por fim, aplicando o algoritmo à imagem pretendida recorrendo ao código *readtext*. Para testar o algoritmo, foi utilizada a imagem presente na Figura 25.

Figura 25: Imagem utilizada para extração de texto

Capítulo 4

Resultados

Este capítulo apresenta os resultados obtidos para cada categoria, isto é, as matrizes de confusão e métricas de avaliação dos modelos. Cada categoria contém três sub secções de resultados, uma para cada uma das abordagens.

É importante referir que na primeira categoria apresentam-se dois cenários diferentes para classificação de texto e imagem de forma a comparar a sua eficácia. Nas restantes categorias apenas os melhores cenários de ambas as abordagens foram apresentados.

A última secção refere-se à extração de texto de uma imagem, abordagem que acabou por ser abandonada por motivos apresentados na mesma.

4.1 Vegetariano

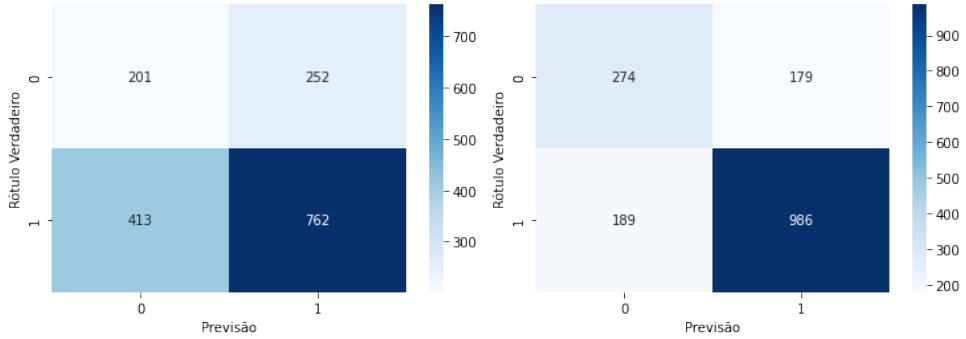
Foram realizados vários cenários para esta abordagem porém. Nesta secção, analisaram-se os dois melhores dado que apresentaram resultados aceitáveis e com informação interessante.

Esta categoria foi utilizada para comparação de cenários, dessa forma apresentam-se e comparam-se dois cenários diferentes, utilizando as arquiteturas *EfficientNet* e VGG-16 como extratoras de características e um SVM como classificador. Na Figura 26 estão duas matrizes, uma para cada cenário citado em cima.

Relativamente ao primeiro cenário, a performance do modelo com a arquitetura *EfficientNet* não foi a esperada e este obteve uma *accuracy* relativamente baixa, apenas 0.5915, e um *F1-score* de 0.6962. Apresentou também falhas na classificação da classe negativa, que podem ser comprovadas pela baixa especificidade de 0.4437. Na classe positiva o modelo mostrou ter melhor capacidade de classificação com valor para a sensiblidade de 0.6485.

Com a arquitetura VGG-16 os resultados obtidos foram melhores face aos anteriores, no que diz respeito à *accuracy* e *F1-score*, os valores obtidos foram 0.7740 e 0.8427, uma melhoria de 0.1825 e

0,1465, respetivamente. Quanto aos valores de sensiblidade e especificidade, este modelo apresentou um valor de 0.8391 e 0.6049.



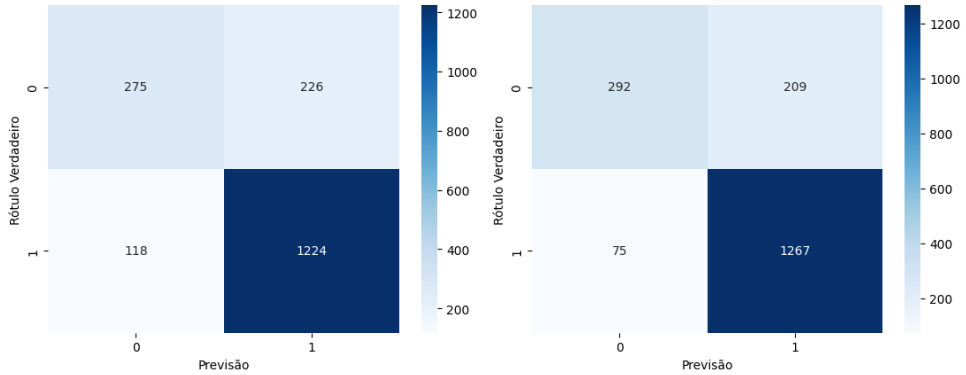
a Cenário com *EfficientNet* b Cenário com VGG-16

Figura 26: Matrizes de confusão do SVM com diferentes arquiteturas como extratoras de características

Na Figura 27a a matriz de confusão obtida através do modelo de classificação de texto utilizando o GloVe e CNN nos dados de teste. O modelo apresentou uma *accuracy* de 0.81, um *F-1 Score* de 0.87. Este mostrou ter uma notável capacidade de classificação da classe positiva (vegetarianas), apresentando uma sensibilidade de 0.9121. Na classe negativa (não vegetarianas), a performance não foi a desejada dado que apresenta erros, mostrado pela baixa especificidade (0.5489).

Na Figura 27b apresenta-se a matriz de confusão utilizando o modelo BERT. Este apresentou uma *accuracy* de 0.8459 e um *F-1 score* de 0.8992. Relativamente à sensibilidade e especificidade, apresenta valores superiores ao modelo anterior, sendo 0.9441 e 0.5828, respetivamente.

Apesar do segundo modelo apresentar uma melhor performance, foram encontrados alguns problemas quando o modelo era carregado de forma independente (sem efetuar o treino previamente), pelo que se optou por seguir com o primeiro modelo.



a Cenário com *GLoVE* e CNN b Cenário com *BERT*

Figura 27: Matrizes de confusão obtidas através da CNN e do modelo BERT

O *ensemble* (ver Figura 28) mostrou uma *accuracy* 0.7065 e um *F-1 score* de 0.7842, resultados piores do que quando se utilizou os modelos em separado. Quanto à sensibilidade e especificidade, este

modelo tomou valores de 0.7325 e 0.6367, respetivamente.

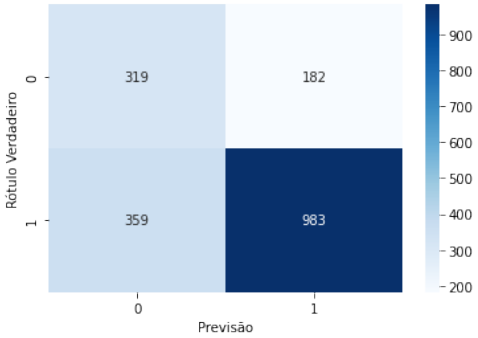


Figura 28: Matriz de confusão *ensemble* Vegetariano

4.2 *Vegan*

Na categoria de *Vegan* os resultados apresentados são apenas com as abordagens escolhidas anteriormente.

Em classificação de imagem (ver Figura 29), o modelo apresentou uma *accuracy* de 0.7352. Este mostrou ter melhor capacidade de classificar a classe positiva do que a negativa, dado que a sua sensibilidade é de 0.7634 e a especificidade de 0.6967. Esta capacidade pode ser explicada pelo facto de na classe positiva não existirem observações duplicadas, isto é, o modelo generaliza melhor a aprendizagem já que não está a treinar com imagens repetidas. Em termos de *F-1 Score*, este modelo apontou um valor de 0.7691

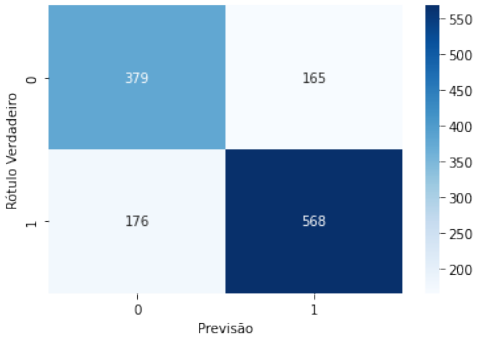


Figura 29: Matriz de confusão classificação de imagem *Vegan*

Em classificação de texto, os resultados obtidos estão presentes na Figura 30. O modelo revelou uma *accuracy* de 0.8328 e *F-1 score* de 0.8652. Mostrou também uma sensibilidade de 0.9038 e especificidade de 0.7290.

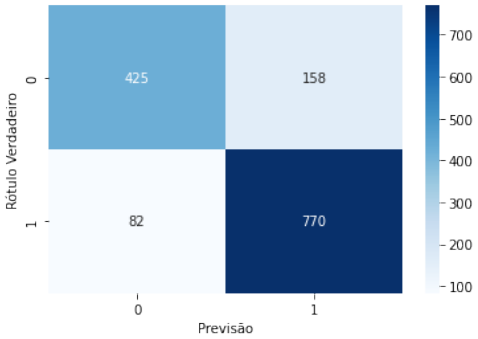


Figura 30: Matriz de confusão classificação de texto *Vegan*

Quanto ao *ensemble* este exibiu uma *accuracy* de 0.7707, melhor que o modelo de classificação de imagem, porém pior que o de classificação de texto. O mesmo aconteceu com o *F-1 score* que apresentou valor de 0.7916.

Quanto à sensibilidade e especificidade este modelo tomou valores de 0.7336 e 0.8250, respetivamente. (Resultados retirados da Matriz de confusão presente na Figura 31)

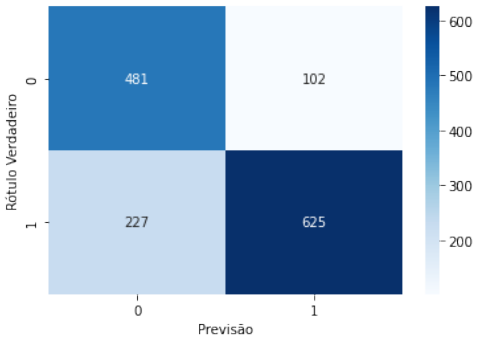


Figura 31: Matriz de confusão *ensemble* Vegan

4.3 Sem lactose

Na categoria de Sem Lactose, é importante referir que classe positiva refere-se a não ter lactose. Em classificação de imagem (resultados na Figura 32), à semelhança das categorias anteriores o modelo apresentou uma performance melhor na classificação da classe positiva, mostrado pela sensibilidade e especificidade de 0.8255 e 0.6260, respetivamente. Este apontou também uma *accuracy* de 0.7545 e um *F-1 score* de 0.8124.

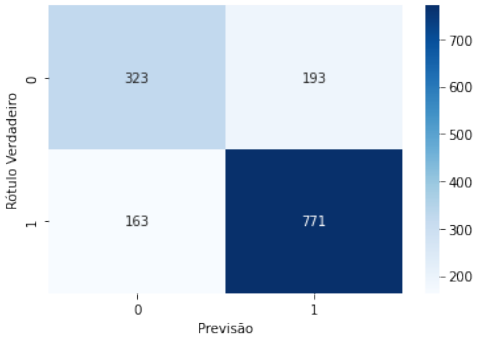


Figura 32: Matriz de confusão classificação de imagem Sem lactose

Na Figura 33, estão presentes os resultados obtidos com o modelo de classificação de texto. Apresentou uma *accuracy* de 0.8396, um *F-1 score* de 0.8822, bem como uma sensibilidade 0.9282 e especificidade de 0.6772.

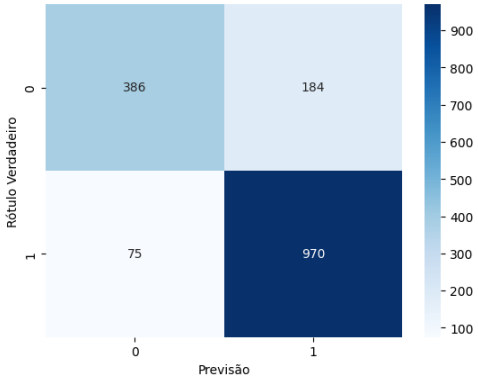


Figura 33: Matriz de confusão classificação de texto Sem lactose

No *ensemble*, verificou-se uma melhoria na especificidade face às duas abordagens em separado, sendo este valor de 0.7298, porém no que toca à sensibilidade, esta piora face à classificação de texto, tomando o valor de 0.8201. No que toca às principais métricas, apresentou o valor de 0.7882 para a *accuracy* e 0.8337 para o *F-1 score*.

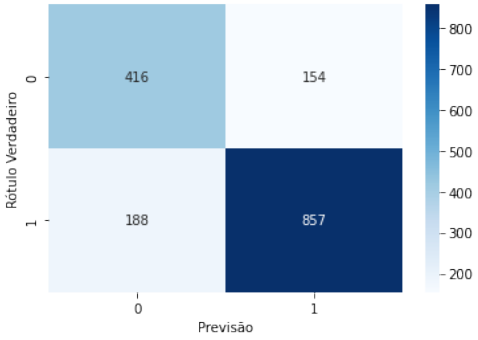


Figura 34: Matriz de confusão *ensemble* Sem lactose

4.4 Sem glúten

Em relação à presente categoria e à semelhança da anterior, a classe positiva refere-se a não conter Glúten. A performance do modelo de classificação de imagem, está apresentada na forma de matriz de confusão na Figura 35. Dessa matriz foram retiradas algumas métricas para ajudar na avaliação do modelo. Este apresentou uma *accuracy* 0.7313, bem como um *F-1 score* de 0.7856. À semelhança das categorias anteriores, apresenta uma melhor capacidade de classificar a classe positiva, justificada pelo valor da sensibilidade ser superior ao da especificidade, 0.7852 e 0.6406, respetivamente.

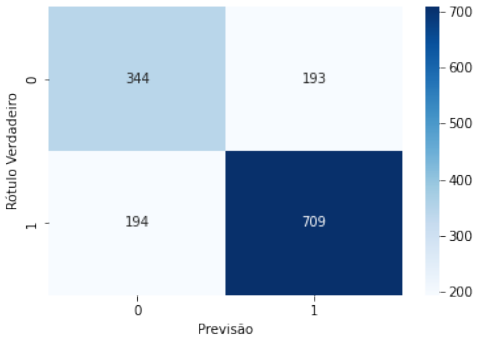


Figura 35: Matriz de confusão classificação de imagem Sem glúten

No modelo de classificação de texto, este demonstrou uma *accuracy* e *F-1 score* de 0.7930 e 0.8450, respetivamente. Quanto à sensibilidade e especificidade, estas apresentaram valores de 0.8996 e 0.6137, o que evidência uma melhor capacidade do modelo para prever a classe positiva.

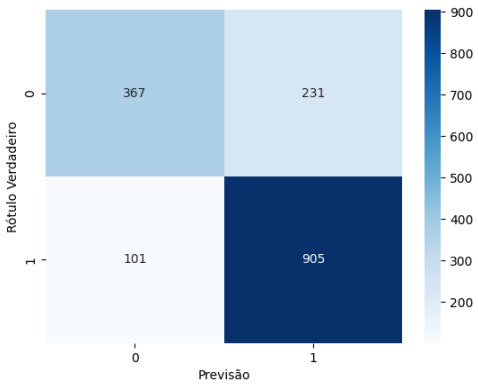


Figura 36: Matriz de confusão classificação de texto Sem glúten

Para o *ensemble* verificou-se uma diminuição da *accuracy* e *F-1 score* face aos dois modelos anteriores, tendo assumido valores de 0.7219 e 0.7630, respetivamente. A sensibilidade do modelo tomou o valor de 0.7137 e a especificidade de 0.7358.

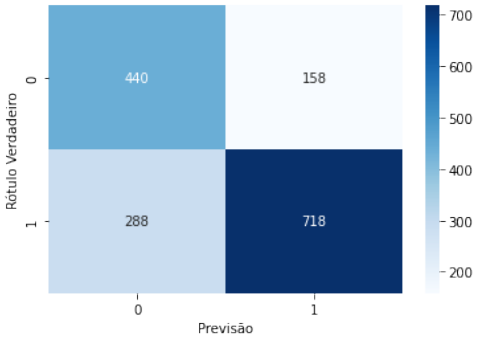


Figura 37: Matriz de confusão *ensemble* Sem glúten

4.5 Sem açúcar

Nesta categoria a classe positiva é referente à ausência de açucares adicionados no produto.

Em relação a classificação de imagem o modelo apontou uma *accuracy* de 0.7369 e um *F-1 score* de 0.7466. No que toca à sensibilidade e especificidade, este apresentou valores de 0.7694 e 0.7040.

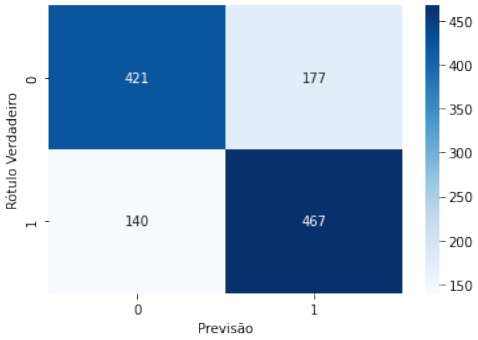


Figura 38: Matriz de confusão classificação de imagem Sem açúcar

Na classificação de texto o modelo mostrou uma *accuracy* de 0.7846 e um *F-1 score* de 0.8077. Face ao modelo de classificação de imagem, este apresentou uma sensibilidade bastante superior, com valor de 0.8823, porém um valor ligeiramente mais baixo para a especificidade, 0.6820.

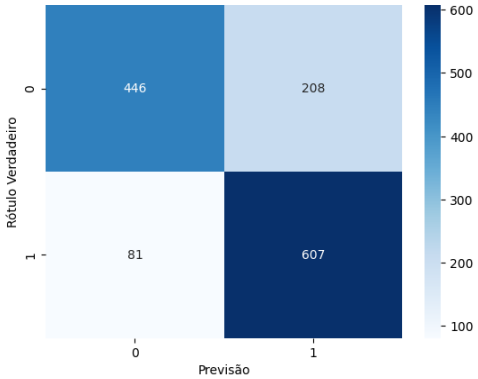


Figura 39: Matriz de confusão classificação de texto Sem açúcar

O modelo de *ensemble* exibiu uma *accuracy* e *F-1 score* de 0.8055 e 0.8093, respetivamente. No que toca a sensibilidade o valor que o modelo apresentou foi 0.8052 e especificidade foi 0.8058.

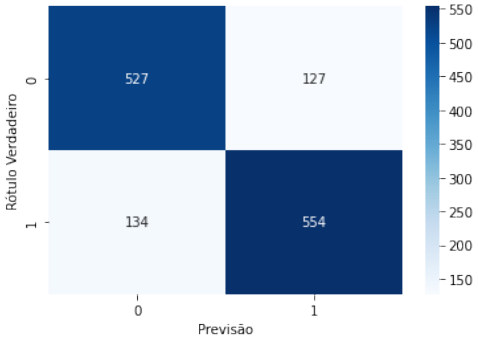


Figura 40: Matriz de confusão *ensemble* Sem açúcar

4.6 Biológico

Na categoria de Biológico, o modelo de classificação de imagem apresentou uma *acurracy* de 0.8719 e um *F-1 score* de 0.7586. Este apresentou também uma sensibilidade de 0.7571 e uma especificidade de 0.9135.

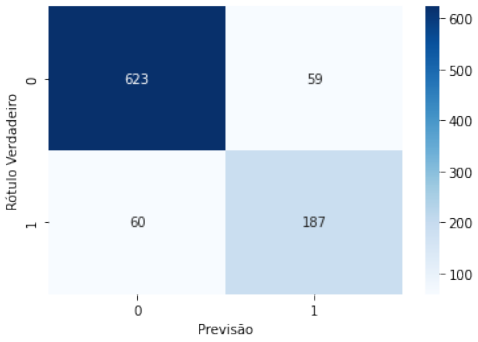


Figura 41: Matriz de confusão classificação de imagem Biológico

Na abordagem de classificação de texto, o modelo apresentou valores excelentes, atingiu uma *accuracy* de 0.9710 e um *F-1 score* de 0.9389, bem como uma sensibilidade e uma especificidade de 0.9630 e 0.9734, respetivamente.

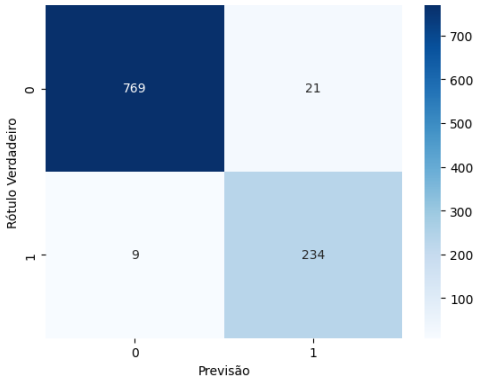


Figura 42: Matriz de confusão classificação de texto Biológico

No *ensemble*, apesar de piorar ligeiramente face ao modelo de classificação de texto, este apresenta valores interessantes. Quanto a *accuracy* e *F-1 score*, este tomou valores de 0.9485 e 0.8987, respetivamente. A sensibilidade tomou o valor de 0.9004 e a especificidade de 0.9648.

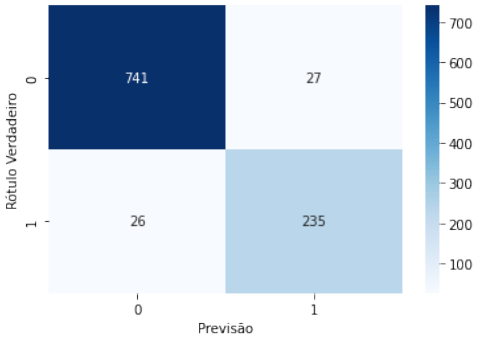


Figura 43: Matriz de confusão *ensemble* Biológico

4.7 Extração de texto de uma imagem

Esta abordagem, como mencionado anteriormente, seria utilizada como um complemento à classificação de texto, tentando acrescentar mais informações que pudessem ser úteis ao modelo. Foi utilizada como teste a imagem presente na Figura 44.

Na Figura 45 está presente o resultado obtido. É possível verificar que, apesar da deteção de palavras existentes na imagem, foram também detetados *tokens* sem qualquer sentido. O tempo de processamento também era elevado, cerca de três minutos por imagem, o que tornava esta abordagem impossível de ampliar para a totalidade dos produtos. Dessa forma esta abordagem acabou por ser abandonada uma vez que os custos de aplicar este algoritmo seriam enormes e o ganho relativamente baixo.

Figura 44: Imagem utilizada para a extração de texto de uma imagem

Nestle	Daoppear(buuos
Bolero	CQjiCEeuruld
CEREAIS	MivUiMim
E FIBRA	SEM
CORgHe	AÇÚCARES
Weucou	ADICONADOs *
Energial	Disiiis
PortuGaL	'Con(v
[25à|	Auns

Figura 45: Resultado obtido com a utilização do OCR

4.8 Discussão

Em geral os modelos apresentaram resultados positivos, com a exceção do vegetariano. Há uma melhor performance por parte dos modelo de texto, algo expectável uma vez que a informação fornecida ao modelo era mais concreta enquanto que nas imagens está presente mais "ruído".

Na classificação de imagem a escolha entre qual arquitetura utilizar como extratora de características foi simples. Os resultados obtidos com a arquitetura *EfficientNet* eram piores que os obtidos com a arquitetura VGG-16. Posto isto, optou-se por utilizar a VGG-16.

A performance dos modelos de imagem foi, de forma geral, satisfatória e notou-se uma capacidade de classificar corretamente a classe positiva face à classe negativa com a exceção da categoria de Biológico. Esta tendência pode ser explicada pelo facto do número de observações positivas ser superior ao das negativas. Foi necessária a utilização do *oversampling* para igualar o número de observações, o que dificultou a generalização da aprendizagem do modelo uma vez que os dados de treino possuíam imagens "repetidas" da classe negativa.

Na classificação de texto a escolha do modelo não foi baseada apenas da performance, apesar do modelo BERT apresentar um rendimento superior à CNN, deparou-se com problemas ao tentar carregar o mesmo depois de treinado. Dessa forma, optou-se por continuar com a CNN.

O desempenho dos modelos de classificação de texto foi, de forma geral, positiva . À semelhança dos modelos de classificação de imagem, a sensibilidade dos modelos é superior à especificidade, conclusão que pode ser explicada pelo do facto do texto conter menos ruído que a imagem, hipótese mencionada na abordagem anterior. Na categoria de Biológico, a performance do modelo de classificação de texto foi excelente, explicada pelo facto dos produtos biológicos apresentarem a palavra "bio" nas suas descrições (marca, sub-marca, produtor, etc.), decorrente da forte aposta estratégica da empresa nesta tendência/-necessidade alimentar, e mostrado pela sua *accuracy* de 0.9710.

No *ensemble*, os resultados apesar de positivos, não foram os esperados. O objetivo da criação dos *ensembles* era unir as duas abordagens de forma a tirar proveito dos pontos fortes de cada uma e apesar de, geralmente, este apresentar resultados mais razoáveis que uma das abordagens isoladas, não atingiu o objetivo principal.

Por fim, os resultados da extração de texto de uma imagem não foram os esperados. Apesar de efetivamente ter sido retirada alguma informação útil sobre o produto, na sua maioria consistia em informação de pouca relevância, adicionando o facto do tempo de processamento ser demasiado elevado, foi uma abordagem que acabou por não ser utilizada.

Capítulo 5

Conclusões

5.1 Conclusões

Nesta dissertação foram desenvolvidos diferentes algoritmos de classificação automática de produtos do retalho alimentar por imagem, texto e uma junção dos dois anteriores. Estes algoritmos tiveram por base métodos de ML. Foi também aplicado um algoritmo de extração de texto de uma imagem com auxilio a técnicas de OCR.

Foram realizadas diversas reuniões com diferentes equipas da empresa para perceber que dados estavam disponíveis para ajudar no desenvolvimento deste projeto. Disponibilizaram-se duas bases de dados diferentes, um banco de dados do Continente *on-line* com imagens de produtos e um *excel* que continha informações/descrições dos mesmos, onde alguns apresentavam pré-classificações nas categorias de vegetariano, *vegan*, sem açúcar, sem glúten, sem lactose e biológico. Com um cruzamento dos dados foi possível obter os dados necessários para efetuar o treino e teste dos modelos de imagem e texto.

Em classificação de imagem, o processo de treino dos modelos foi muito demorado, principalmente porque a primeira abordagem, utilizando as arquiteturas *EfficientNet* e VGG-16 como classificadoras, devolveu resultados insatisfatórios. Investiu-se bastante tempo no *fine tunning* do modelo para tentar obter melhores resultados, sem sucesso. Foi então que se abandonou a utilização das arquiteturas como classificadoras e se utilizou apenas como extratoras de características, posteriormente alimentadas a um SVM para a classificação.

Para a classificação de texto foram testadas duas abordagens, uma com uma CNN e *embeddings* GloVe e outra utilizando o modelo BERT. Ambas forneceram resultados favoráveis mas a decisão de qual abordagem selecionar baseou-se no facto do modelo BERT ter apresentado problemas quando o modelo era carregado posteriormente.

O *ensemble* foi criado com base numa regressão logística, onde os dados de entrada seriam as probabilidades de classificação dos modelos de imagem e texto. Os resultados obtidos, apesar de não serem melhores que os resultados de ambos os modelos anteriores individualmente, não foram, de todo, instatisfatórios. O *ensemble* na categoria de vegetariano foi uma surpresa pela negativa, uma vez que apresentou uma performance bastante baixa face aos modelos individuais.

Em geral os modelos das três abordagens mencionadas tendencialmente classificam melhor a classe positiva do que a classe negativa, devido ao facto de no treino estes terem mais diversidade de informação nessa classe (positiva), o que facilita a generalização da aprendizagem.

Na extração de texto de uma imagem, os resultados obtidos não foram os esperados. Encontraram-se bastantes problemas de compatibilidade de bibliotecas de OCR, o que dificultou a tarefa em termos de tempo e qualidade da extração. Posteriormente quando encontrada a biblioteca *easy OCR* foi possível identificar palavras na imagem porém muitas delas não possuíam nenhum sentido gramatical. Esta biblioteca apresentou um outro problema, sendo este o tempo de processamento de cerca de três minutos por imagem, o que inviabilizava a aplicação para toda a gama de produtos. Com base nestes problemas, esta abordagem acabou por ser abandonada uma vez que não seria viável nas condições disponíveis.

5.2 Trabalho Futuro

No contexto de melhorar o trabalho desenvolvido, é indispensável melhorar a performance das abordagens.

Na classificação de imagem uma sugestão seria treinar um modelo com uma base de dados diferente, onde continha produtos com os *stickers* referentes à categoria em que se inserem e posteriormente usar esse modelo como classificador ou extrator de características. Analisar a qualidade dos dados seria outra sugestão uma vez que foram encontrados casos de produtos mal pré-classificados que podem ter induzido o modelo atual em erro em alguns casos. Finalmente poderiam ser testados mais hiperparâmetros e arquiteturas diferentes, algo que não foi possível devido às limitações computacionais.

Na classificação de texto seria importante à semelhança da abordagem anterior, rever a qualidade dos dados de forma a reduzir ao máximo dados mal pré-classificados e testar diferentes arquiteturas. A criação de um novo modelo de classificação de texto onde os dados de entrada seriam, ao invés de descrições do produto, marca, etc., a lista de ingredientes do produto. Este modelo poderia ser mais uma ajuda para o *ensemble*.

Seria interessante testar diferentes formas de *ensembles*, com a junção de mais modelos poderia ser testado por exemplo um *ensemble* de votação.

De forma a tentar tornar a abordagem da extração de texto de imagem viável, seria importante testar diferentes bibliotecas, como por exemplo a *Tesseract OCR*, que atualmente apresenta erros de compatibilidade com o ambiente.

Bibliografia

[1] M. Awad e R. Khanna. *Efficient learning machines: theories, concepts, and applications for engineers and system designers*. Springer nature, 2015 (ver p. 3).

[2] F. Chollet. *Deep learning with Python*. Simon e Schuster, 2021 (ver p. 3).

[3] A. Abraham. "Artificial neural networks". Em: *Handbook of measuring system design* (2005) (ver pp. 5, 9).

[4] R. E. Wright. "Logistic regression." Em: (1995) (ver p. 5).

[5] R. Burbidge e B. Buxton. "An introduction to support vector machines for data mining". Em: *Keynote papers, young OR12* (2001), pp. 3–15 (ver p. 5).

[6] A. Mammone, M. Turchi e N. Cristianini. "Support vector machines". Em: *WIREs Computational Statistics* 1.3 (2009), pp. 283–289. doi: https://doi.org/10.1002/wics.49. eprint: https://wires.onlinelibrary.wiley.com/doi/pdf/10.1002/wics.49. url: https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wics.49 (ver p. 6).

[7] S. Sahu, M. Prasad e B. Tripathy. "A support vector machine binary classification and image segmentation of remote sensing data of Chilika Lagloon". Em: *International Journal of Research in Information Technology* 3.5 (2015), pp. 191–204 (ver p. 6).

[8] G. Anthony, H. Gregg e M. Tshilidzi. "Image classification using SVMs: one-against-one vs one-against-all". Em: *arXiv preprint arXiv:0711.2914* (2007) (ver p. 6).

[9] M. Awad e R. Khanna. *Efficient Learning Machines*. 2015. Cap. 1. doi: https://doi.org/10.1007/978-1-4302-5990-9 (ver p. 9).

[10] K. O'Shea e R. Nash. "An Introduction to Convolutional Neural Networks". Em: (2015). url: http://arxiv.org/abs/1511.08458 (ver p. 9).

[11] S. Dang e P. H. Ahmad. "Text mining: Techniques and its application". Em: *International Journal of Engineering & Technology Innovations* 1.4 (2014), pp. 22–25 (ver pp. 11, 12).

[12] S. V. Gaikwad, A. Chaugule e P. Patil. "Text mining methods and techniques". Em: *International Journal of Computer Applications* 85.17 (2014) (ver pp. 11, 12).

[13] M. Allahyari et al. "A brief survey of text mining: Classification, clustering and extraction techniques". Em: *arXiv preprint arXiv:1707.02919* (2017) (ver p. 11).

[14] A. K. Uysal e S. Gunal. “The impact of preprocessing on text classification”. Em: *Information Processing Management* 50.1 (2014), pp. 104–112. issn: 0306-4573. doi: https://doi.org/10.1016/j.ipm.2013.08.006. url: https://www.sciencedirect.com/science/article/pii/S0306457313000964 (ver p. 12).

[15] S. Vijayarani, M. J. Ilamathi, M. Nithya et al. “Preprocessing techniques for text mining-an overview”. Em: *International Journal of Computer Science & Communication Networks* 5.1 (2015), pp. 7–16 (ver p. 12).

[16] K. Kowsari et al. “Text classification algorithms: A survey”. Em: *Information* 10.4 (2019), p. 150 (ver p. 13).

[17] M. Ikonomakis, S. Kotsiantis e V. Tampakas. “Text classification using machine learning techniques.” Em: *WSEAS transactions on computers* 4.8 (2005), pp. 966–974 (ver p. 13).

[18] M. D. Deepa et al. “Bidirectional encoder representations from transformers (BERT) language model for sentiment analysis task”. Em: *Turkish Journal of Computer and Mathematics Education (TURCOMAT)* 12.7 (2021), pp. 1708–1721 (ver p. 14).

[19] Q. Zhu e J. Luo. “Generative Pre-Trained Transformer for Design Concept Generation: An Exploration”. Em: *Proceedings of the Design Society* 2 (2022), pp. 1825–1834. doi: 10.1017/pds.2022.185 (ver p. 15).

[20] Y. Wei et al. “Deep learning for retail product recognition: Challenges and techniques”. Em: *Computational intelligence and neuroscience* 2020 (2020) (ver pp. 16, 17).

[21] P. Jiang et al. “A Review of Yolo algorithm developments”. Em: *Procedia Computer Science* 199 (2022), pp. 1066–1073 (ver p. 17).

[22] J. Redmon et al. “You only look once: Unified, real-time object detection”. Em: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788 (ver p. 17).

[23] W. Liu et al. “Ssd: Single shot multibox detector”. Em: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer. 2016, pp. 21–37 (ver p. 17).

[24] H. Wang. “Garbage Recognition and Classification System Based on Convolutional Neural Network VGG16”. Em: *2020 3rd International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE)*. 2020, pp. 252–255. doi: 10.1109/AEMCSE50948.2020.00061 (ver p. 18).

[25] A. Sarkar. *Understanding EfficientNet-the most powerful CNN architecture*. 2021 (ver p. 18).

[26] M. Tan e Q. Le. “Efficientnet: Rethinking model scaling for convolutional neural networks”. Em: *International conference on machine learning*. PMLR. 2019, pp. 6105–6114 (ver p. 18).

[27] C. Szegedy et al. “Rethinking the inception architecture for computer vision”. Em: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826 (ver p. 18).

[28] A. Singh, K. Bacchuwar e A. Bhasin. “A survey of OCR applications”. Em: *International Journal of Machine Learning and Computing* 2.3 (2012), p. 314 (ver p. 19).

[29] T. Drabas e D. Lee. *Learning PySpark*. Packt Publishing Ltd, 2017 (ver p. 21).

[30] M. Nosrati. “Python: An appropriate language for real world programming”. Em: *World Applied Programming* 1.2 (2011), pp. 110–117 (ver p. 22).

[31] url: https://www.python.org/doc/essays/blurb/ (ver p. 22).

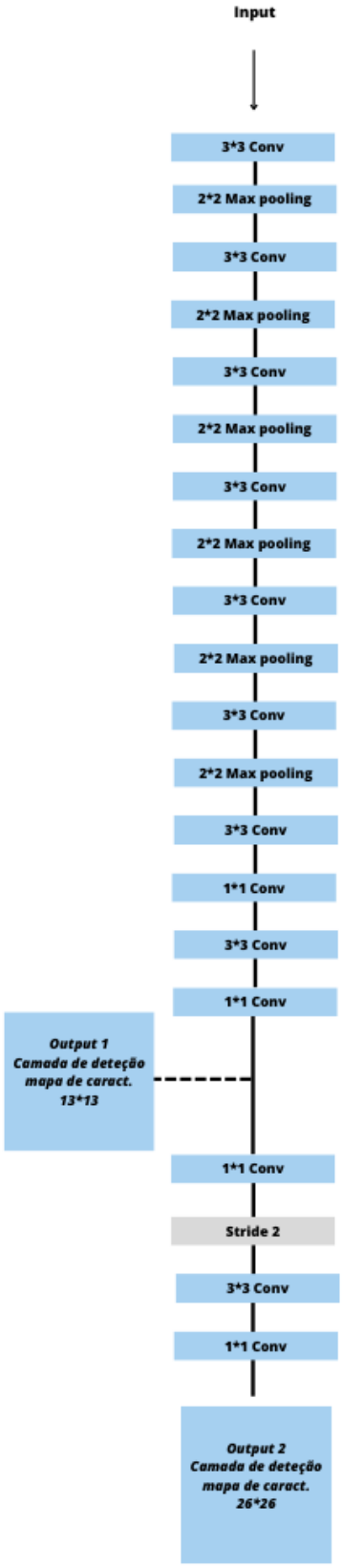
[32] S. Staff. *What is SQL (structured query language)? - SQLCOURSE*. 2022-02. url: https://www.sqlcourse.com/beginner-course/what-is-sql/ (ver p. 22).

[33] *re - Operações com expressões regulares*. https://docs.python.org/pt-br/3/library/re.html. Accessed: 2023-08-25 (ver p. 32).

Anexo I

Anexos

I.1 Arquitetura YOLOv3 *tiny*


Input
3*3 Conv
2*2 Max pooling
3*3 Conv
2*2 Max pooling
3*3 Conv
2*2 Max pooling
3*3 Conv
2*2 Max pooling
3*3 Conv
2*2 Max pooling
3*3 Conv
2*2 Max pooling
3*3 Conv
1*1 Conv
3*3 Conv
1*1 Conv
Output 1
Camada de deteção
mapa de caract.
13*13
1*1 Conv
Stride 2
3*3 Conv
1*1 Conv
Output 2
Camada de deteção
mapa de caract.
26*26