Report do Projeto Final: Aprendizagem Automática Aplicada

Conceito de uma Aplicação – Ajudar os consumidores nas compras

Categoria do projeto: Classificação de Imagem

Miguel Ventura Gomes Paulo, 25432

Instituto Superior de Agronomia, Mestrado em Green Data Science 2022-2023

Introdução

Neste projeto idealizei uma aplicação, podendo ser para dispositivos móveis, onde o utilizador tivesse no seu bolso uma ferramenta para o auxiliar no momento de fazer as suas compras em supermercados ou outras lojas do retalho alimentar.

Especialmente em Portugal, esta aplicação seria de grande utilidade, pois permitiria ao consumidor, se quisesse poupar dinheiro ou comprar os produtos de maior qualidade do mercado, aceder a um banco de dados onde tivesse toda a informação sobre um produto em específico, como o histórico de preços, locais onde esse produto está a ser vendido mais barato, os "pros" e os "contra" desse produto ou mesmo uma classificação dada por outros utilizadores da aplicação.

Para isso, recorri à Classificação de Imagem para identificar o produto através de uma foto tirada pelo utilizador com o seu telemóvel, identificando o produto, facilmente lhe seriam apresentadas todas as informações relevantes sobre ele.

Dados

Para a classificação de imagem, foi necessário ter acesso a um banco de imagens de cada um dos produtos que eu estava interessado em introduzir no modelo para realizar o projeto. Como não foi possível encontrar e ter acesso a essas imagens, recorri a uma forma menos eficaz, porém muito eficiente para ter algumas imagens para testar o meu modelo.

Assim sendo, utilizei em Python a biblioteca Fast.ai, que me permitiu fazer grande parte do modelo, incluindo também a recolha de imagens da internet. Para o fazer o utilizei, com recurso ao Fast.ai, uma função que me ia procurar e fazer download das imagens correspondentes àquela pesquisa. Essa função chama-se: "*search_images_ddg*", que basicamente usa um motor de busca chamado "*DuckDuckGo*" para obter as imagens.

```
if not path.exists():
    path.mkdir()
    for o in products:
        dest = (path/o)
        dest.mkdir(exist_ok=True)
        results = fa.search_images_ddg(f'{o}')
        fa.download_images(dest, urls=results)
```

Imagem 1: Função "*search_images_ddg*"

Como se observa na Imagem 1, podemos ver de que forma a função é usada, porém também que temos uma lista de produtos que a função deve usar, chamada de "*products*", nessa lista coloquei os produtos alimentares que me interessaram: Compal Maracujá 1L, Nesquik Chocolate em pó, Atum Bom Petisco, Iogurte Aroma de Morango Mimosa, Queijo Flamengo Limiano, Bolacha Maria Vieira e por fim Bolacha Oreo Original.

Indicando os produtos, a função já é capaz de pesquisar exatamente o que eu lhe pedi, e com a função "*download_images*" consegui armazenar tudo em "*path*" o que a função encontrou na Internet.

Imagem 2: Exemplos de Imagens encontradas e descarregadas pelo código da Imagem 1

A função foi capaz de armazenar cerca de 200 imagens de cada produto, sendo suficiente para treinar o modelo para qualquer fotografia tirada pelo utilizador.

A maior parte dos dados necessários já estão armazenados, porém como pretendo também dar alguma informação relevante após a identificação do produto, tive de criar 2 ficheiros csv com dados sobre os tópicos que eu falei em cima, mais especificamente o histórico de preços e a classificação atual do produto.

Para isso fui à procura dos preços médios mensais destes produtos em 2022 e após encontrar a informação, coloquei tudo num ficheiro csv. Para as classificações, como é um termo que não existe, tive de criar dados do zero sem nenhuma base e também criei um ficheiro csv com esses dados.

Explicação do Modelo de Classificação de Imagem

Após recolher todos os dados necessários (Imagens + dados para a informação adicional), começou-se a criar o modelo, para isso primeiramente teve que se ter a certeza que todas as imagens recolhidas não estão corrompidas, para isso usou-se a função "*verify_images()*", que basicamente verifica imagem por imagem, sendo posteriormente excluídas todas as imagens inválidas com recurso à função "*path.unlink*".

```
1  failed = fa.verify_images(fns)
```

```
1  failed.map(fa.Path.unlink)
```

Imagens 3 e 4: Aplicação das funções (fns representa a localização das imagens)

Após esse processo, já temos todas as imagens normalizadas, ou seja, teremos cerca de 200 imagens por produto que serão válidas para ensinar o modelo.

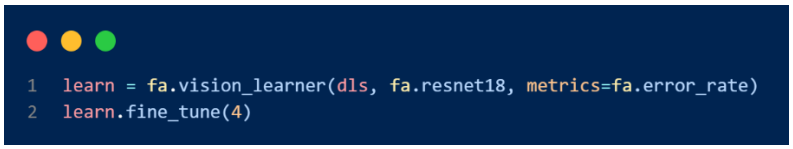
De seguida chegou o momento de ensinar o modelo, como não temos uma grande variedade de imagens usou-se a função "*RandomResizedCrop()*" com uma imagem de 224 pixéis.

Esta função realiza um recorte aleatório na imagem original, mantendo a proporção, e redimensiona o recorte para o tamanho que defini em cima. Esta função é usada para aumentar a diversidade do conjunto de dados de treino.

Também é usada o módulo *"aug_transforms()"*, que oferece uma variedade de transformações, como rotação, espelha a imagem, faz cortes aleatórios, ajusta o brilho e o contraste, entre outros.

Imagem 5: Função *"RandomResizedCrop()"* e *"aug_transforms()"*

E por fim, usou-se a função "*vision_learner()*" para que o modelo aprendesse com as transformações feitas em cima. Mais à frente também se exportou o modelo para se obter resultados mais adiante.



Imagens 6 e 7: Aplicação da função *"vision_learner()"* e como se exportou o modelo

Resultados

Com o modelo criado, já é possível obter-se alguns resultados. Começando pela eficácia do modelo, com a *Confusion Matrix* obtida, podemos concluir que o modelo é bastante eficaz, tendo um *Error rate* bastante baixo.

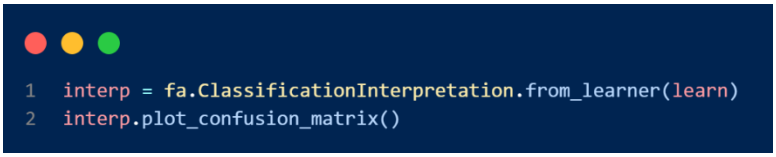


Imagem 8: Criação da *Confusion Matrix*

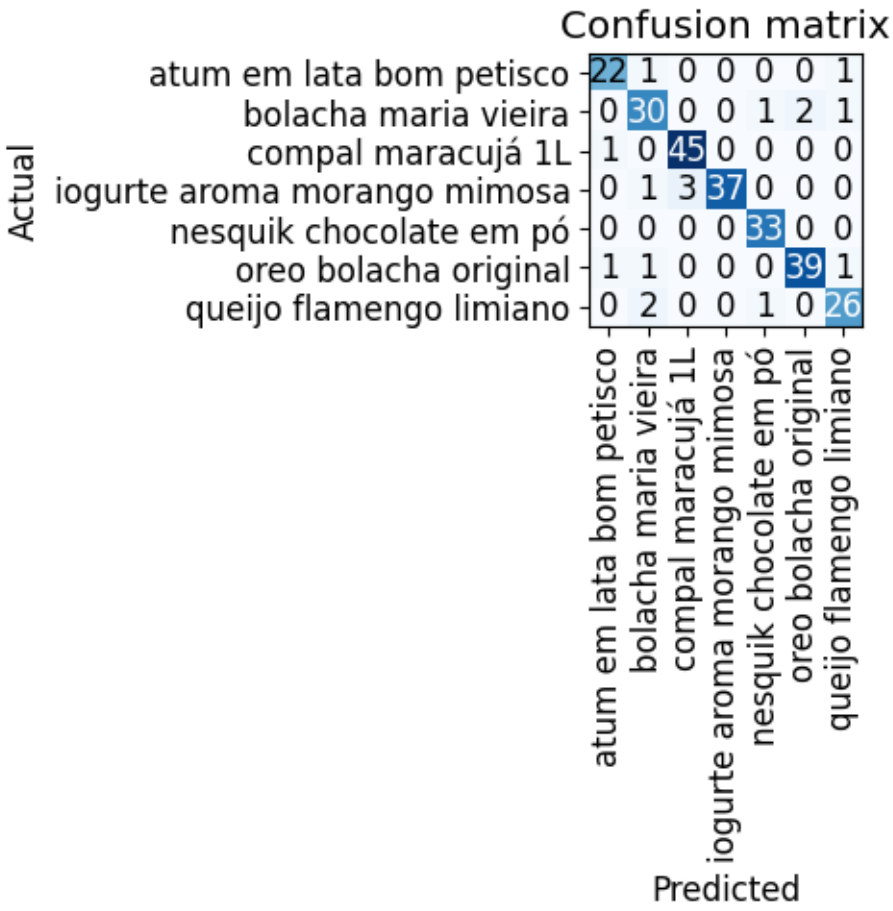


Imagem 9: *Confusion Matrix* obtida

Sabendo que o modelo criado é bastante viável, foi-se testar o modelo com uma imagem que não foi incluída durante a aprendizagem do modelo. Desta forma, conseguimos observar se o modelo acerta no tipo de produto e se nos dá a informação relativa a esse mesmo produto.

Assim, começou-se por importar o modelo e prever a categoria de uma imagem que eu forneci ao modelo.


```
learn_inf = fa.load_learner('model_produtos.pkl')
```

```
produto_previsto = learn_inf.predict('Compal.jpg')[0]
print(produto_previsto)
```

Imagens 10,11 e 12: Produto testado (Compal.jpg), *Import* do modelo e Previsão da categoria

Como esperado, o nosso *Output* foi: “compal maracujá 1L”, que corresponde ao nome da categoria de todos os produtos que são o Compal de Maracujá de 1L.

Desta forma, confirma-se que o modelo está a responder corretamente.

Como prometido, também foi desenvolvida uma forma de o consumidor receber informação adicional quando o produto fosse identificado.

Para isso, tive de recorrer a 2 ficheiros csv, onde está toda a informação para o histórico de preços e para a classificação de cada produto.

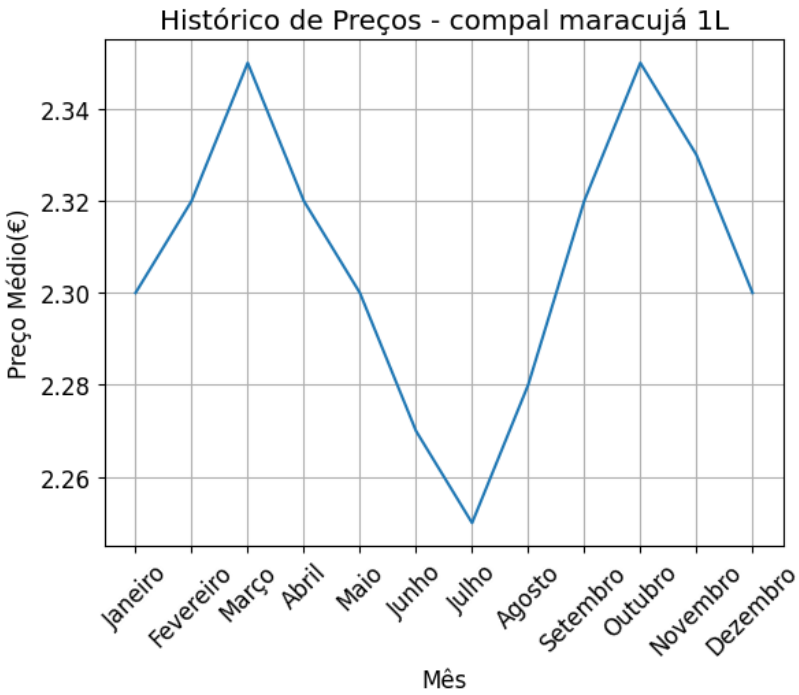
De forma a ir buscar a informação que eu queria aos meus ficheiros, tive de criar uma condição com *IF* para que o código saiba em que coluna está cada informação dependendo do produto que foi previsto. Na imagem 13, pode-se ver o processo que foi feito para o histórico de preços, mas também para a classificação.

```python
import pandas as pd
precos = pd.read_csv('historico_de_precos.csv', encoding='utf-8')

if produto_previsto == 'atum em lata bom petisco':
    precos_produto_previsto = precos['preco_atum_bompetisco']
```

Imagem 13: Condição para produto_previsto == ‘atum em lata bom petisco’

Finalizando a análise dos resultados, com tudo definido, foi possível criar um gráfico e uma linha de *output* com a informação disponível sobre o Compal de Maracujá.




```
A classificação do produto compal maracujá 1L é 4.6 ⭐
```

Imagens 14 e 15: Gráfico – Histórico de Preços do Compal de Maracujá 1L e o *Output* da classificação do produto

Discussão e Conclusões

O objetivo deste trabalho foi desenvolver um modelo de classificação de imagem utilizando um modelo de Machine Learning para identificar produtos específicos a partir de 7 categorias diferentes de alimentos com base em imagens. O modelo foi treinado com um conjunto de dados extenso e diversificado, que continha as várias categorias de produtos.

Foram utilizadas diversas técnicas de processamento de imagem para extrair características relevantes das imagens e ensinar o modelo de classificação. Além disso, foram realizadas várias etapas de pré-processamento, como redimensionamento, normalização e aumento de dados, para garantir que o modelo fosse capaz de lidar com diferentes variações das imagens.

Durante a fase de treino, foram aplicadas técnicas de validação para avaliar o desempenho do modelo. A precisão alcançada foi bastante satisfatória como foi visto na *Confusion Matrix.*

No entanto, alguns desafios foram identificados durante o processo. Por exemplo, certos produtos podem ter aparências semelhantes, o que dificulta a distinção precisa entre as categorias. Além disso, variações na iluminação, ângulo de visão e qualidade da imagem podem afetar o desempenho do modelo.

Apesar disso, o maior desafio foi mesmo a recolha de imagens, tendo que utilizar o Fast.ai para descarregar e identificar as imagens, onde uma parte delas, devido à variedade de marcas, acabou-se por treinar o modelo com produtos de marcas diferentes, porém contendo o mesmo alimento.

De qualquer forma a melhoria do desempenho pode ser obtida com a expansão do conjunto de dados de treino ou a utilização de uma modelo mais preciso e eficaz.

No futuro, o modelo poderia ser implementado num conceito de aplicativo móvel ou sistema de reconhecimento de alimentos em supermercados e lojas, contribuindo para uma experiência de compra mais ágil, fácil e automatizada.

Referências

Notebook para a criação do modelo de classificação de imagem: greends-pml /Lesson2_edited_book_02_production.ipynb. Disponível no Github da disciplina. Última vez consultado em: 8 Junho 2023

Criação dos dados com Faker, Disponível em: https://pypi.org/project/Faker/0.7.4/, Última vez consultado em: 8 Junho 2023