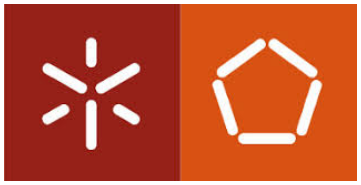

Universidade do Minho
Mestrado em Engenharia Informática

Aprendizagem Profunda

Previsão da idade cerebral a partir da sua conectividade estrutural

Grupo 8

Carolina Santejo	Filipa Pereira	Luís Pinto	Raquel Costa
pg47102	pg46978	pg47428	pg47600

Braga
13 de Maio de 2022

Conteúdo

1 Introdução e Contextualização

Nos últimos tempos, com o envelhecimento da população tem-se verificado um aumento da prevalência de doenças neurodegenerativas. O envelhecimento é caracterizado pelas alterações tanto do cérebro como da cognição, que têm um grande grau de variância dependendo do indivíduo. É de notar que esta variância é causada por inúmeros fatores, tais como genética, estilo de vida, etc.

De modo a possibilitar o diagnóstico precoce destas doenças, têm sido desenvolvidos, ultimamente, modelos de aprendizagem profunda capazes de prever a idade cerebral de um indivíduo. Esta estratégia permite que seja possível calcular a diferença entre a idade prevista do cérebro e a idade cronológica da pessoa. Assim sendo, quando esta diferença é maior do que zero é possível concluir que existe um envelhecimento cognitivo acelerado.

2 Principais Objetivos

O principal objetivo deste trabalho consiste no desenvolvimento de um modelo de *Deep Learning* capaz de prever a idade do cérebro de indivíduos a partir da sua representação matricial de conectividade e de outros dados complementares sendo eles o seu género e grau de literacia. Por outro lado, pretende-se também determinar, no conjunto de dados de treino, quais as pessoas que poderão padecer de alguma patologia, sendo estas identificadas pela diferença significativa entre a sua idade biológica e a idade cerebral . Além disto, no âmbito desta UC, realizou-se uma competição na plataforma *Kaggle* na qual é suposto utilizar o modelo desenvolvido para prever as idades do cérebro de indivíduos a partir de dados de um ficheiro de teste. Pretende-se assim, que a média absoluta do erro entre as idades reais e previstas seja a menor possível, evitando *overfitting*.

3 Metodologia

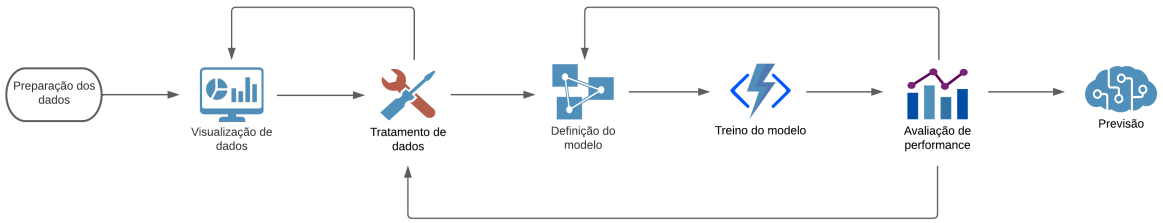

Figura 1: Metodologia adotada

Para desenvolver um modelo de aprendizagem profunda capaz de prever a idade do cérebro de indivíduos, o grupo seguiu uma metodologia bem definida constituída por 7 fases. De uma forma geral, analisando a figura 1 pode-se observar que este processo

nem sempre é sequencial, ou seja, existem fases onde é necessário regressar a etapas anteriores para conseguir melhores resultados.

Assim sendo, de uma forma mais detalhada a metodologia é constituída pelos seguintes passos:

Preparação dos dados: Nesta fase são carregados os dados dos ficheiros para memória passando os mesmos para estruturas de dados bem definidas.

Visualização de dados: Esta fase tem como objetivo analisar os dados fornecidos, ou seja, é nesta etapa que se pretende encontrar valores inconsistentes ou em falta, relações entre atributos, identificar *outliers*, avaliar a qualidade dos dados, etc.

Tratamento de dados: Após a visualização de dados é necessário efetuar o tratamento dos mesmos. Nesta fase são escolhidos as *features* relevantes para o modelo, serão completadas ou retiradas entradas constituídas por valores em falta e, caso haja valores inconsistentes é necessário corrigi-los. Nesta etapa é também fundamental transformar todos os atributos com tipos de dados incompatíveis com o modelo de aprendizagem para tipos compatíveis, utilizando as técnicas correspondentes a cada um.

Definição do modelo: Nesta etapa é construída a rede neuronal a utilizar. Desta forma, são determinadas todas as camadas da rede, funções de ativação correspondentes e outros parâmetros necessários.

Treino do modelo: Tendo a rede neuronal construída é necessário treiná-la com os dados fornecidos. Assim sendo, em primeiro lugar, são definidas métricas de avaliação e outros hiperparâmetros tais como o número de épocas. Por fim, são fornecidos os dados de treino à rede para desta forma ser treinada.

Avaliação de performance: Para verificar o sucesso do treino da rede é feita uma avaliação da sua performance. Consequentemente, são então comparados os dados obtidos com os esperados, tanto dados de treino como de validação, para que relacionando o erro entre os dois tipos se possa identificar situações de *overfitting* ou *underfitting*.

Previsão: Quando a avaliação do modelo é positiva, pode ser feita a previsão utilizando a rede treinada. Nesta fase são fornecidos dados nunca vistos pela rede anteriormente, sendo que esta deve prever com o máximo de precisão o seu *output*.

4 Descrição e Exploração dos dados

Para desenvolver um modelo de aprendizagem profunda foram fornecidos dois tipos de datasets: um ficheiro MAT com os dados das imagens da conectividade do cérebro e um ficheiro CSV onde para cada pessoa são indicadas as suas informações pessoais, tais como a idade, género e nível de educação. Ambos os ficheiros foram carregados para memória, sendo que o primeiro foi guardado numa matriz de 3 dimensões (112x90x90) e o segundo carregado para um dataframe.

Primeiramente, analisando os dados do dataframe foi possível concluir que não existem valores em falta nas colunas fornecidas. De seguida foi analisado o seu *pairplot* onde se pode observar a relação entre os diversos atributos. Do mesmo modo, foi também analisada a distribuição dos dados de cada coluna do dataframe.

De seguida, o grupo passou à análise dos dados da matriz da conectividade do cérebro. Deste modo foi gerada uma representação visual de cada imagem, onde se pode observar as zonas com maior (branco) e menor (preto) conectividade. Em adição, é também visível a existência de uma simetria relativamente à diagonal, ou seja, a informação está duplicada para posições simétricas da matriz.

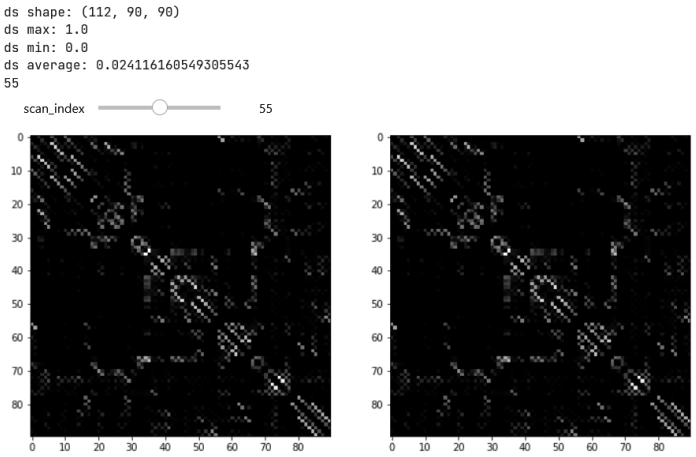

Figura 2: Visualização da matriz de conetividade

Por fim, considerou-se relevante identificar as regiões com maior conexão cerebral dadas as imagens fornecidas. Desta forma, foi feita uma nova matriz onde cada posição corresponde à soma dos valores desse mesmo ponto em todas as matrizes de conectividade. Tendo a matriz das somas representou-se visualmente numa imagem [Fig.3] os seus valores sendo que as zonas com maior conexão estão identificadas a amarelo e as de menor com a cor azul escura. Analisando a figura pode concluir-se que maior parte dos valores da matriz são nulos, o que significa que existem muitas zonas onde não há qualquer conexão cerebral.

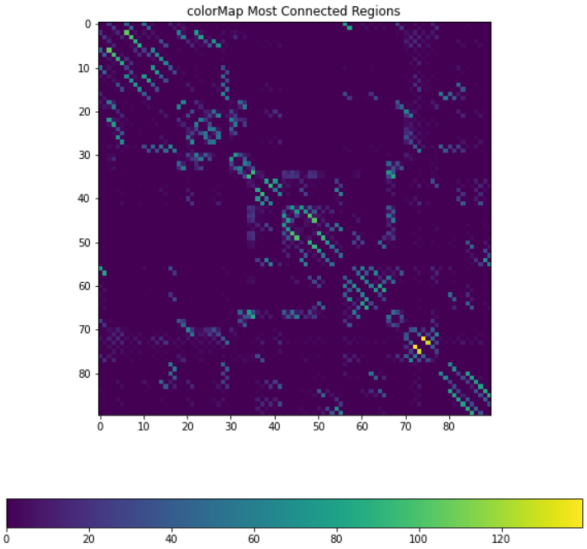

Figura 3: Regiões com maior conexão

5 Tratamento dos dados

Tendo por base a descrição e visualização dos dados, chegamos agora à etapa de tratamento dos mesmos. Assim sendo, o tratamento efetuado consistiu nas seguintes abordagens : a **Normalização das Colunas**, o ***One-Hot Encoding*** e a **conversão de Matrizes para Vetores**.

Em particular, comecemos pela normalização das colunas. Este procedimento foi realizado de forma a normalizar o impacto que os diferentes atributos tem no treino, tentando assim atingir a aprendizagem do modelo de forma menos enviesada. Na prática, isto levou a passar os valores da **educação** do ficheiro train.csv e test.csv para uma escala entre 0 e 1, invés de entre 0 e 20.

As redes neuronais são sensíveis à gama de valores dos atributos, sendo por isso necessário re-escalar os valores dos atributos e transformar variáveis categóricas para números. Isto leva-nos à segunda abordagem adotada, o One-Hot Encoding. Nesta situação, o procedimento foi efetuado nas colunas ***sex*** dos ficheiro train.csv e test.csv, servindo o propósito de transformar a coluna *sex* em duas novas colunas, respetivamente ***Male*** e ***Female***. Esta é uma técnica bastante conhecida, a sua utilidade consiste em conferir uma maior expressividade ao dados e, novamente, de minimizar o peso relativo dos atributos. Neste caso em particular, a solução implementada permitiu resolver a questão de ao existir apenas uma coluna a representar o sexo do participante, onde as linhas com valor 1 teriam uma maior ponderação relativa na rede comparativamente ao valor 0.

Por último, a etapa provavelmente mais importante do processo de tratamento foi a **transformação dos dados em matriz**, dos ficheiros train_data.mat e test_data.mat, **para dados em vetor**. Esta decisão advém do facto de não estar contida uma imagem visual propriamente dita nas matrizes mas sim, a informação relevante, estar contida nos próprios dados matriciais. Em conjugação com o facto de as matrizes serem esparsas, fez ainda mais sentido o processo de transformação dos dados da forma matricial para vetores com apenas informação útil e formatada para o treino. Para chegar ao vetor final foi necessário : eliminar a matriz triangular superior devido a existir informação duplicada com a matriz triangular inferior, eliminar a diagonal pois apresenta informação não importante para o estudo da conexão entre zonas do cérebro e eliminar os zeros comuns a todas as matrizes, ou seja, remover as zonas do cérebro sem ligação. Deste modo, passamos de matrizes 90x90 com 8100 dados, para meias matrizes sem diagonal (90x89)/2 com 4005 dados e finalmente chegamos a vetores sem zeros comuns de 1202 dados. Traduzindo esta taxa de redução no tamanho dos dados de input obtemos um vetor com uma quantidade de dados cerca de **6.74 vezes menor** que o original.

6 Desenvolvimento do Modelo

6.1 Arquitetura do Modelo

Em primeiro lugar, é necessário abordar a arquitetura que o grupo construiu para o modelo desenvolvido. Este encontra-se divido em duas partes, uma CNN e uma MLP. As CNN (Convolutional neural networks) tratam-se de modelos de deep learning nos quais as várias camadas da rede consistem em camadas convolucionais e de "pooling", que recebem inputs e geram outputs. É de notar que cada camada funciona como uma espécie de filtro da camada anterior. Daí o grupo ter decidido usar uma CNN como uma forma de extrair as características dos dados de input, tal como representado na figura [4]. Além disso, este tipo de modelo permite a aprendizagem de forma hierárquica, em que as camadas sucessivamente aprendem conceitos cada vez mais complexos e abstratos. As redes neuronais convolucionais são tipicamente usadas em imagens, que são tratadas como uma matriz de píxeis. Os padrões de aprendizagem são reconhecidos nas várias partes da matriz. É importante ainda referir que as CNN's são muito usadas em campos como o de "Computação de Imagens na Medicina", onde é possível fazer análises preditivas, mais uma razão para o grupo ter escolhido este modelo, visto o tema que abordado no presente trabalho. Em relação ao modelo MLP (Multilayer Perceptrons), trata-se de um modelo composto por séries de camadas totalmente conectadas.

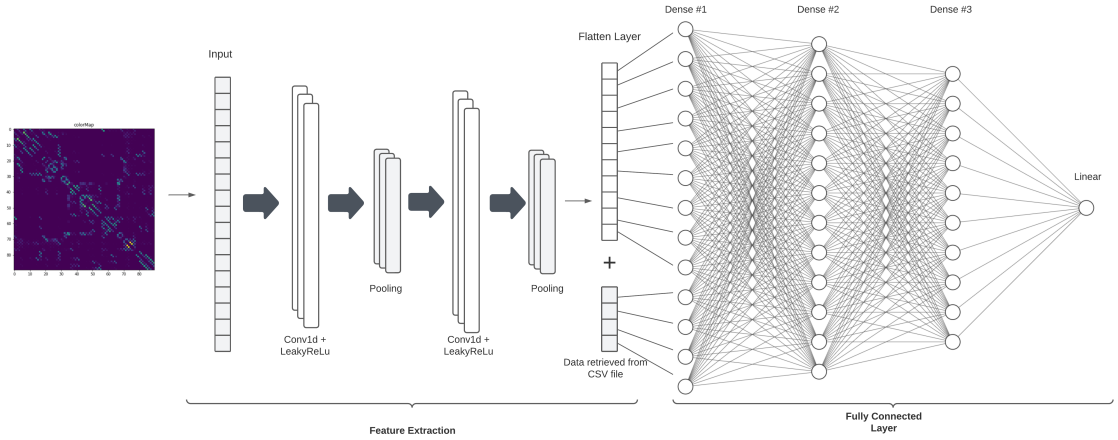

Figura 4: Arquitetura do Modelo Desenvolvido

O primeiro passo do grupo foi converter os dados da matriz de conetividade num vetor, tal como foi referido durante a etapa de "Tratamento de Dados". De seguida, para extrair as principais características do vetor, foram definidas camadas convolucionais tanto "conv1D"como de "pooling". No final desta etapa, depois da última camada de "pooling", é realizado um *flattening*, onde é obtido um vetor que irá servir de input para a MLP. Uma vez que foram fornecidos também ficheiros csv com atributos extra para o nosso modelo, é nesta fase, depois de obter a "Flatten Layer", que será feita a concatenação do vetor resultante da *feature extraction* com o vetor que contém todos os dados que estavam armazenados no ficheiro csv. O vetor final obtido servirá como input da rede neuronal, que contém camadas totalmente conectadas, e que por último irá devolver a previsão da idade cerebral do indivíduo.

6.2 Características e Otimizações

De forma a melhorar o desempenho do modelo de aprendizagem foram definidas algumas caraterísticas e implementadas diversas otimizações de modo a obter os melhores resultados. É de notar que para construir a rede neuronal foi utilizado o módulo *keras*.

Em primeiro lugar, relativamente à complexidade do modelo, como já foi referido, dividiu-se a rede em duas partes: uma CNN e uma MLP. Em relação à rede convolucional foram definidas 4 camadas sendo que se escolheram filtros de 128, 90, 64 e 32 respetivamente. Relativamente à MLP, esta é constituída por 4 camadas com filtros 90, 30, 15 e 1 respetivamente. Estas caraterísticas foram escolhidas após várias tentativas com o cuidado do modelo não ser demasiado complexo, evitando assim o *overfit*. Uma vez que se forneceram vetores como input da rede, tanto as camadas convolucionais como as MaxPool foram utilizadas para uma dimensão, ou seja, **Conv1D** e **MaxPool1D**.

De seguida, fez-se o *tunning* do modelo ajustando os hiperparâmetros. Quanto às funções de ativação, foi utilizada a 'LeakyReLU' uma vez que permite obter melhores resultados do que a 'Relu'. Na camada de output utilizou-se a função de ativação *linear* porque estamos perante um problema de regressão. Relativamente ao kernel size foram testados valores ímpares como 3 ou 5, visto que normalmente são os que se adequam melhor às redes neuronais convolucionais. No final foi escolhido o tamanho 5 porque se obteram melhores resultados, além de que as matrizes de entrada são de grandes dimensões . Da mesma forma utilizou-se o *padding* 'same' para 'aproveitar' todos os atributos da imagem (nomeadamente os cantos) que é extraída pela CNN.

De um modo geral, as redes neuronais têm tendência a overfit. Por este motivo, para diminuir o erro e evitar overfit adicionou-se 'Dropout's após as camadas 'Dense' sendo que com estes se eliminou 30% dos nodos da rede em cada uma. Para além disso, foram também testados modelos com BatchNormalization, no entanto, visto que a rede não é muito complexa, o seu uso não permite não obter melhorias nos resultados.

```
image_input = Input((1202,1))

x = Conv1D(filters = 128, kernel_size = 5 , activation ='LeakyReLU',padding = 'same')(image_input)
x = Conv1D(filters = 90, kernel_size = 5 , activation ='LeakyReLU',padding = 'same')(x)
x = MaxPool1D(pool_size= 2)(x)

x = Conv1D(filters = 64, kernel_size = 5 , activation ='LeakyReLU',padding = 'same')(x)
x = Conv1D(filters = 32, kernel_size = 5 , activation ='LeakyReLU',padding = 'same')(x)
x = MaxPool1D(pool_size= 2)(x)

x = Flatten()(x)

vector_input = Input((3,))
yl = Concatenate()([x, vector_input])

z = Dense(90, activation = "LeakyReLU")(yl)
z = Dropout(0.3)(z)

z = Dense(30, activation = "LeakyReLU")(z)
z = Dropout(0.3)(z)

z = Dense(15, activation = "LeakyReLU")(z)

output = Dense(1,activation = "linear")(z)

model = tf.keras.Model(inputs=[image_input, vector_input], outputs=output)
```

Figura 5: Código da rede neuronal

6.3 Processo de treino

Tendo-se explicado as características e algumas otimizações aplicadas ao modelo, irá agora ser abordado o processo de treino do mesmo. O primeiro passo deste processo consiste em compilar o modelo (usa-se a função *compile* do keras), sendo que para isto definiu-se a função de custo e a função de otimização. Relativamente à *loss function* escolheu-se a **MAE**, *Mean Absolute Error*, que indica a média da diferença absoluta entre os valores previstos e os valores reais da regressão. Escolheu-se esta métrica não só porque seria a utilizada para avaliar a performance dos modelos na competição do Kaggle mas também porque é a métrica mais direta e intuitiva para avaliar o erro de idade que o modelo está a cometer. Isto quer dizer que, neste cenário, não faria sentido utilizar, por exemplo a MSE , pois esta eleva o erro cometido ao quadrado, sendo mais penalizante para o modelo e, por isso, irrealista. Por outro lado, e relativamente ao algoritmo de otimização, foi escolhido o **Adam** (Adaptive Moment Estimation),uma vez que dos vários algoritmos de otimização disponíveis, este é dos que possui melhor performance (converge mais rápido para um mínimo),consegue lidar com dados que possuam *noise*, além de ser computacionalmente eficiente [1] . É de realçar que o grupo escolheu um *lerning rate* inicial de 0.005, dado que verificou que começando com valores mais altos, o *val_loss* ao longo das épocas de treino reduzia de forma mais significativa. Posto isto, é necessário fazer *fit* do modelo, sendo que para isto passa-se como argumentos os dados de treino (lista contendo os vetores de conectividade e dados de educação e género), o *target* (a idade) e os dados de validação. É de realçar que para efetuar a divisão dos dados, o grupo criou uma função específica para o efeito, guardando 10% para validação e 90% para treino.Para além disto, neste processo, mantivemos ainda o *batch_size* de 32, e utilizamos 100 épocas de treino. Também foi invocamos uma *callback* do tipo *ReduceLROnPlateau* que reduz o *learning rate* quando a métrica *val_loss* deixa de diminuir. Nas figuras 6 e 7 é possível observar de forma mais clara este processo de treino. Relativamente à figura 6 é possível observar que a curva de treino vai descendo de forma significativa até á época 60, sendo que a partir da dia desce mais subtilmente e acaba por estabilizar. Já a curva de validação encontra-se bastante coincidente com a curva de treino, sendo que também sofre oscilações significativas até à época 60, estabilizando daí para a frente. Por outro lado,a figura 7 demonstra a variação do *learning rate* (provocada pela *callback*) ao longo das épocas, e podemos observar que este valor diminui duas vezes, sendo que a primeira descida ocorre por volta da época 70, ou seja, quando as curvas de treino e validação já não apresentam grandes oscilações.

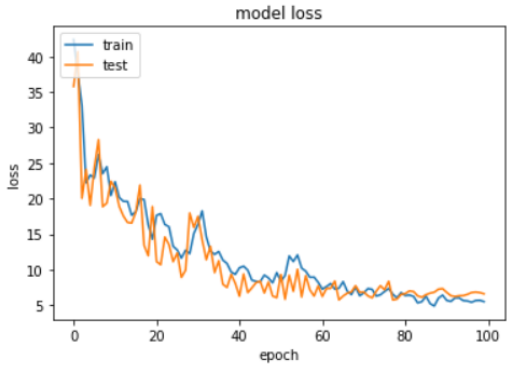

Figura 6: Curvas de treino e validação

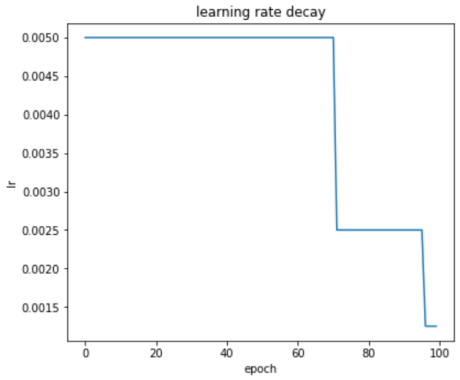

Figura 7: Variação do *learning rate*

7 Análise Crítica dos Resultados Obtidos

7.1 Identificação dos casos mais difíceis de prever

Após o desenvolvimento e otimização do modelo, foi pedido que se identificasse os casos mais difíceis de prever, ou seja, os indivíduos que sofrem de algum tipo patologia e por isso a diferença entre a sua idade biológica e a sua idade cerebral é bastante significativa. Para tal, o primeiro passo consistiu em dividir os 112 registos em treino, validação e teste (utilizando a função definida para o efeito e já referida anteriormente) sendo que os dados de treino e validação seriam usados para o processo de treino do modelo, e os dados de teste serviriam para identificar possíveis *outliers* através da comparação entre as idades calculadas pelo modelo e as idades reais. Ora, este processo de divisão dos dados e identificação do *outliers* no batch de teste foi repetido várias vezes, alterando-se o random_state da função *train_test_split* para se obter distribuições de dados diferentes e assim identificar mais pessoas com patologias cerebrais. Tendo-se feito isto, verificou-se que os indivíduos que mais vezes surgiam com um grande *gap* de idades eram os que possuiam *id* 1, 4, 7, 8 e 12. É interessante também realçar que o grupo desenvolveu um pequeno código em Python para calcular o número de desconexões cerebrais de cada pessoa, e , tal como se vê na figura 8, os *outliers* detetados possuem um número de zonas do cérebro desconectadas superior à média (linha vermelha do gráfico).

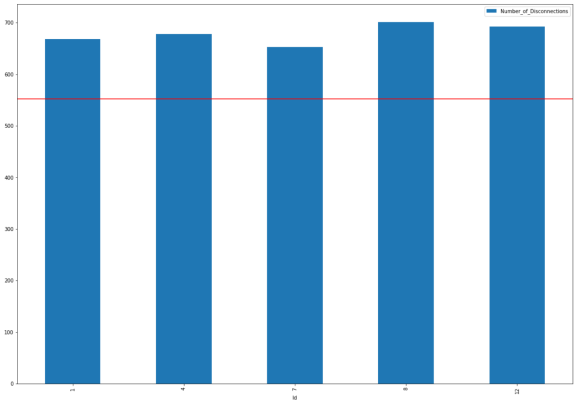

Figura 8: Numero de desconexões cerebrais dos outliers

7.2 Conexões cerebrais mais relevantes

Com este trabalho prático, foi também pedido ao grupo que abordasse a *interpretabilidade* do modelo, ou seja, determinar que conexões cerebrais (features) tiveram maior impacto no modelo desenvolvido. Esta informação é de elevada relevância, pois a partir desta é possível identificar quais as conexões que poderão estar relacionadas com o aparecimento de patologias. Ora, para determinar a *feature importance* existem diversas bibliotecas disponíveis, nomeadamente o SHAP[2] ou o ELI5[3]. O grupo decidiu usar a biblioteca ELI5, recorrendo ao método de **Permutation Importance**. Este método consiste essencialmente em analisar de que forma a ausência de dadas *features* impacta a métrica de avaliação do modelo, ou seja, que conexões cerebrais, quando ausentes, provocam o aumento significativo da *loss*. Assim sendo, podemos observar pela tabela da imagem 9, as vinte conexões mais impactantes.

Weight	Feature
0.1818 ± 0.0796	70 71
0.1054 ± 0.1794	70 76
0.0969 ± 0.1193	9 73
0.0899 ± 0.0749	34 35
0.0786 ± 0.0835	21 25
0.0780 ± 0.0831	9 75
0.0742 ± 0.0405	5 75
0.0712 ± 0.0741	5 73
0.0697 ± 0.0249	1 77
0.0639 ± 0.0474	4 74
0.0549 ± 0.0501	57 77
0.0482 ± 0.0179	25 73
0.0458 ± 0.0764	9 71
0.0453 ± 0.0226	5 83
0.0445 ± 0.2723	75 77
0.0386 ± 0.0344	20 30
0.0347 ± 0.0426	20 82
0.0337 ± 0.0284	36 76
0.0317 ± 0.0269	9 77
0.0296 ± 0.0249	20 72
… 1182 more …	

Figura 9: Importância das Conexões Cerebrais

8 Conclusão e Trabalhos Futuros

Dado por concluído o projeto, fará sentido apresentar uma visão crítica, refletida e ponderada do trabalho realizado.

No espetro positivo, consideramos relevante destacar o facto de existir uma boa visualização e tratamento dos dados, que potencializa a CNN desenvolvida e por sua vez o modelo preditivo. Em adição, foi tido especial cuidado com o *overfitting*, tendo-se obtido uma boa distribuição de idades no ficheiro de previsão.

Em termos de melhorias do projeto, consideramos que seria excelente conseguir diminuir ainda mais o MAE do modelo através de mais tentativas e testes ao *tunning* dos parâmetros da nossa CNN, recorrendo, por exemplo, ao *GridSearch*.

Salientamos ainda que durante a sua pesquisa para a realização do projeto, o grupo adquiriu novos conhecimentos pertinentes, como o uso de *callbacks* para adequar gradualmente a taxa de aprendizagem do modelo e o uso de ferramentas para a interpretabilidade de modelos *black-box*.

Num trabalho futuro, seria interessante explorar a utilização de um *Autoencoder* de forma a escolher as melhores *features* ou, porventura, experimentar uma estratégia com base em *Graph Neural Network* capaz modelar as regiões do cérebro e as suas conexões.

Em suma, consideramos que o balanço do trabalho realizado é positivo, as dificuldades sentidas foram superadas, os requisitos propostos foram cumpridos e os aspetos a melhorar podem ser facilmente atingidos num trabalho futuro.

Referências

[1] Adam Optimization algorithms in Deep Learning, https://www.tech-quantum.com/adam-optimization-algorithms-in-deep-learning/

[2] Convolutional Neural Networks cheatsheet, https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks,Afshine Amidi (2022)

[3] Welcome to the SHAP documentation, https://shap.readthedocs.io/en/latest/index.html

[4] Welcome to ELI5's documentation!, https://eli5.readthedocs.io/en/latest/blackbox/permutation_importance.html

[5] Convolutional Neural Networks (CNN), https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-4-full-connection, SuperDataScience, (2022)