Universidade do Minho
Escola de Engenharia
Mestrado em Engenharia Informática

Dados e Aprendizagem Automática

Ano Letivo de 2024/2025

Conceção e otimização de modelos de Machine Learning

António Silva(pg57867) Duarte Leitão(pg57872) Vasco Faria(pg57905)

21 de janeiro de 2025

DAA

Índice

1 Introdução

Neste trabalho prático foi-nos pedido para fazer a conceção e a otimização de variados modelos de aprendizagem automática de forma a conseguirmos prever com sucesso a progressão de défice cognitivo leve (MCI) para a Doença de Alzheimer (AD). O MCI representa uma fase intermédia entre o envelhecimento e a demência, sendo um fator de risco para o desenvolvimento da AD. A previsão desses casos pode contribuir para intervenções mais eficazes e melhoria da qualidade de vida das pessoas.

De forma a atingirmos este objetivo, exploramos dois *datasets* de características radiômicas obtidas a partir de imagens de ressonância magnética (MRI) do hipocampo e do lobo occipital. A hipótese inicial é que as características radiômicas do hipocampo, que é uma região crucial para a memória, apresentarão diferenças significativas em relação ao lobo occipital, não associado ao desenvolvimento de demências, logo o hipocampo seria um *dataset* melhor para a previsão que queremos fazer.

Este trabalho foi dividido em duas tarefas:

- Tarefa de **Análise e Validação**: Exploração, análise e preparação dos *datasets*, com a validação da relevância de cada região cerebral para a previsão de progressão do MCI utilizando os modelos desenvolvidos para a Tarefa de **Competição**.
- Tarefa de **Competição**: Desenvolvimento e otimização de modelos de Machine Learning utilizando o *dataset* do hipocampo, com a avaliação de desempenho a ser feita na plataforma **Kaggle**.

A abordagem envolverá a utilização de técnicas estatísticas e de aprendizagem automática para análise dos padrões extraídos das imagens, com o objetivo de criar modelos robustos para a previsão da progressão do MCI para AD.

2 Metodologia

Para o desenvolvimento deste trabalho, foi adotada a metodologia **SEMMA** (*Sample, Explore, Modify, Model, Assess*), uma abordagem sistemática para a criação e validação de modelos de aprendizagem automática. Essa metodologia foi aplicada da seguinte forma neste projeto:

- **Sample (Amostragem)**: inicialmente, pelos professores, foi gerada uma amostra representativa dos dados de características radiômicas das imagens de ressonância magnética (MRI) disponíveis no dataset **DShippo** (hipocampo), que está dividido em dois, um para treino de modelos e outro para teste na tarefa de **Competição**, e no dataset **DSocc** (lobo occipital). A amostragem visou garantir a diversidade e a adequação dos dados para o treinamento e validação dos modelos de machine learning.

- **Explore (Exploração)**: os *datasets* foram explorados para visualização e algumas descrições básicas. Análises estatísticas e de distribuição de algumas variáveis foram realizadas(*Age*, *Sex* e *Transition*), tentando encontrar padrões e comportamentos relevantes, especialmente no contexto das diferenças entre o hipocampo e o lobo occipital.

- **Modify (Modificação)**: as variáveis dos *datasets* foram selecionadas e transformadas conforme necessário, considerando a sua relevância para o problema de previsão de progressão de MCI para AD. Técnicas de normalização e feature selection foram aplicadas para garantir a comparabilidade entre as características extraídas.

- **Model (Modelação)**: utilizamos diversos modelos de aprendizagem automática, incluindo **Decision Tree Classifier e Max Voting Classifier**, entre outros abordados durante o semestre. A otimização dos hiperparâmetros foi realizada usando **GridSearchCV**, com o objetivo de maximizar a performance dos modelos em métricas como o **F1-Score**.

- **Assess (Avaliação)**: A *accuracy* e a utilidade dos modelos foram avaliadas utilizando métricas como **F1-Score** e **precision**. Foi conduzida uma análise crítica dos resultados obtidos, comparando o desempenho entre os conjuntos DShippo e DSocc para validar a hipótese de que o hipocampo tem maior relevância na progressão de MCI para AD e analisamos os resultados dados pelo **Kaggle** para vermos quão bons estavam os nossos modelos na previsão do DShippo.

A metodologia **SEMMA** é algo cíclico e iterativo, permitindo que façamos ajustes e refinamentos ao longo do desenvolvimento do projeto. As suas etapas foram repetidas sempre que necessário, de forma a melhorar a performance e a robustez dos nossos modelos desenvolvidos.

3 Análise e Exploração dos datasets

Antes de fazermos qualquer tratamento dos dados nos *datasets* que nos foram fornecidos, fizemos uma pequena comparação entre o *dataset* **DSHippo** de treino e o *dataset* **DSOcc** para encontramos todas as semelhanças e diferenças entre os dois. Após isso, fizemos uma pequena análise dos dados principais com os quais conseguimos fazer gráficos, pois a maioria dos dados destes *datasets* são de natureza médica e muito difíceis de serem analisados.

3.1 Comparação entre o dataset DSHippo(Train) e o dataset DSOcc

O primeiro passo na comparação entre os dois *datasets* foi verificar a *shape* de cada um, onde conseguimos confirmar que ambos tinham **305 linhas e 2181 colunas**, vendo que potencialmente continham dados sobre os mesmos 305 pacientes. Logo a seguir, vimos os tipos das colunas, confirmando novamente que ambos os *datasets* continham os mesmos tipos de colunas: **2014 float64, 147 int64 e 20 object**.

De seguida, comparamos os nomes das colunas entre os dois datasets para ver se havia alguma diferença de nome devido ao facto de os *datasets* serem sobre partes diferentes do cérebro. Conseguimos conferir novamente que as colunas dos dois datasets têm todas os **mesmos nomes**, apesar de provavelmente terem as suas diferenças.

Para confirmar que se tratam de *datasets* de partes diferentes do cérebro, mas sobre os mesmos pacientes, fizemos uma comparação dos valores de cada coluna entre os dois *datasets*. Guardamos num ficheiro de texto, dividido por tipo de coluna, as colunas com valores iguais. Das 2014 colunas do tipo *float64*, somente 15 colunas dessas continham os mesmos valores nos 305 pacientes, sendo uma percentagem mísera de **0.74%** de semelhança entre essas colunas e, visto que essas colunas são a grande parte dos *datasets*, confirmamos que estes datasets em termos de valores são de facto muito diferentes. Além disso, das 147 colunas do tipo *int64*, nesse caso já 124 são iguais, incluindo a coluna **Sex**, colocando-nos um passo mais próximo de confirmar a nossa hipótese de que são os mesmos pacientes nos dois *datasets*. E, por fim, das 20 colunas do tipo *object*, 15 são iguais, incluindo a coluna *Image*, *ID* e *Transition*, confirmando assim a nossa teoria com certeza de que ambos os *datasets* contêm os mesmos pacientes, pois utilizam a mesma imagem, devendo depois cada um focar na sua parte do cérebro, utilizam o mesmo *ID* e são todos sobre a mesma *Transition*. Para concluir, das 2181 colunas, verificamos então que somente 154 são iguais, sendo a maioria do tipo *int64*.

3.2 Análise dos datasets

Depois desta comparação entre *datasets* e termos verificado que os pacientes realmente eram os mesmos nos dois, realizamos uma pequena análise de certas variáveis, sendo elas a **Age**, **Sex** e **Transition**. Nesta análise, vamos utilizar os *datasets* do hipocampo.

Começamos para ver como se distribuía a **Age** no *dataset* de Treino e no *dataset* de Teste do Hipocampo para ver se tinha uma distribuição mais ou menos similar.

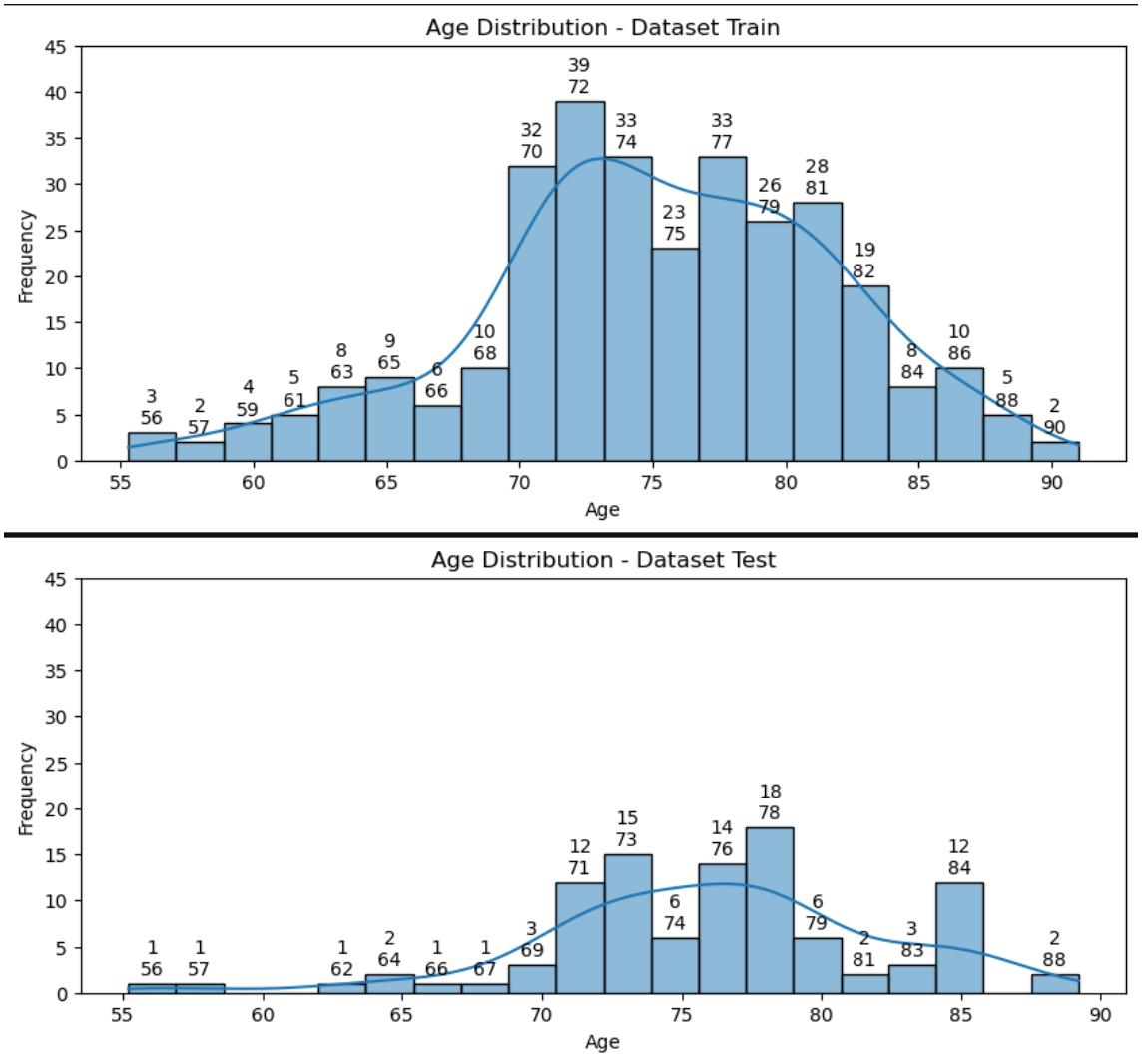

Figura 3.1: Age Distribution in both Hippocampus datasets

Pela linha formada nos dois gráficos conseguimos ver que a distribuição é parecida nos dois *datasets*, apesar de o segundo ter uma quantidade menor de pacientes. Além disso, esta amostra está concentrada em pacientes a partir da idade em qual se começa a encontrar mais MCI, sendo neste *dataset* em concreto no intervalo dos **56 aos 90 anos**.

De seguida, fomos ver a distribuição do **Sex** nos dois *datasets*, onde verificamos que apesar das percentagens serem um bocado diferentes, continuava a ver uma maioria do **Sex** 1, continuando assim a ter essa diferença correta.

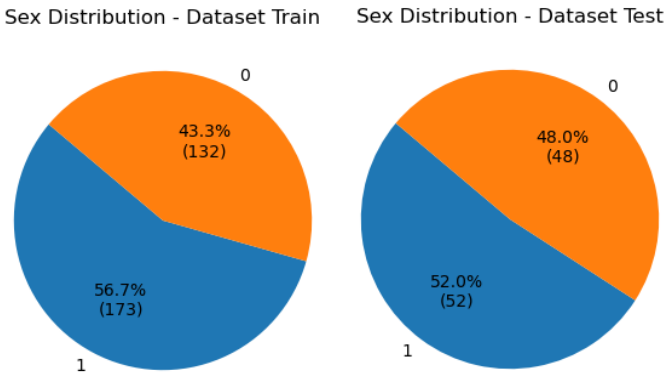

Figura 3.2: Sex Distribution in both Hippocampus datasets

Antes de fazermos uma comparação entre estas variáveis fomos também ver a quantidade de cada transição diferente no *dataset* para ter uma melhor noção com o que estamos a trabalhar para a previsão.

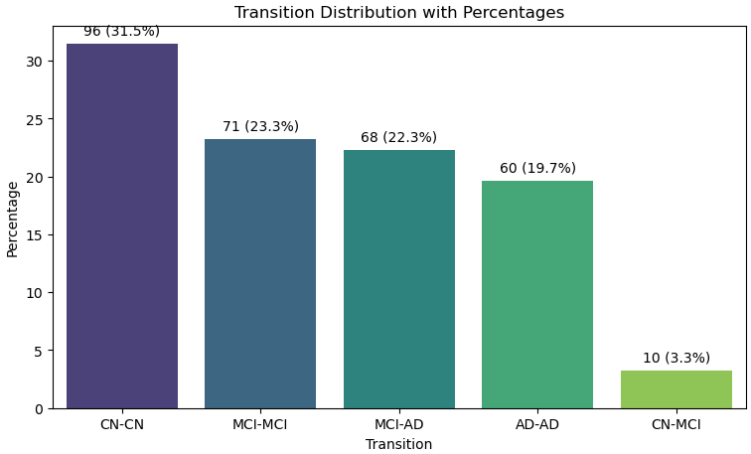

Figura 3.3: Transition Distribution in the Hippocampus Train dataset

Conseguimos ver que a transição que existe em maior quantidade é a CN-CN com quase um terço do *dataset* a ser desse caso e CN-MCI com a menor percentagem com somente 10 casos, o que fará com que provavelmente seja muito complicada de prever essa transição.

Para terminar esta análise dos *datasets* vimos a distribuição de *Age* e *Transition* por *Sex* de forma a ver se existia algum padrão de imparcialidade.

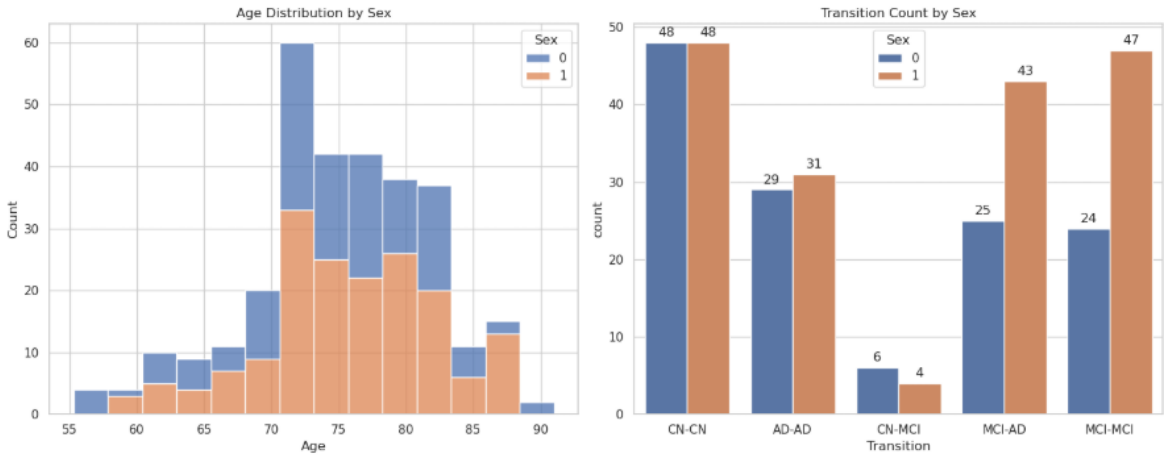

Figura 3.4: Transition Distribution in the Hippocampus Train dataset

Reparamos que em termos de *Age* por *Sex* encontra-se bem distribuído por cada idade sem haver grandes discrepâncias. Já no caso da *Transition* por *Sex* reparamos que no caso de MCI-AD e MCI-MCI existem muitos mais casos do *Sex* 1, mas isso claramente também acontece devido ao facto de existirem cerca de mais 41 pacientes desse *Sex* que neste caso foram parar a essas duas *Transition* em específico.

Além destas comparações, fizemos também um *boxplot* que relaciona estas três variáveis entre si, demonstrando o range de cada Transition na *Age* e como este range está definido em cada *Sex*.

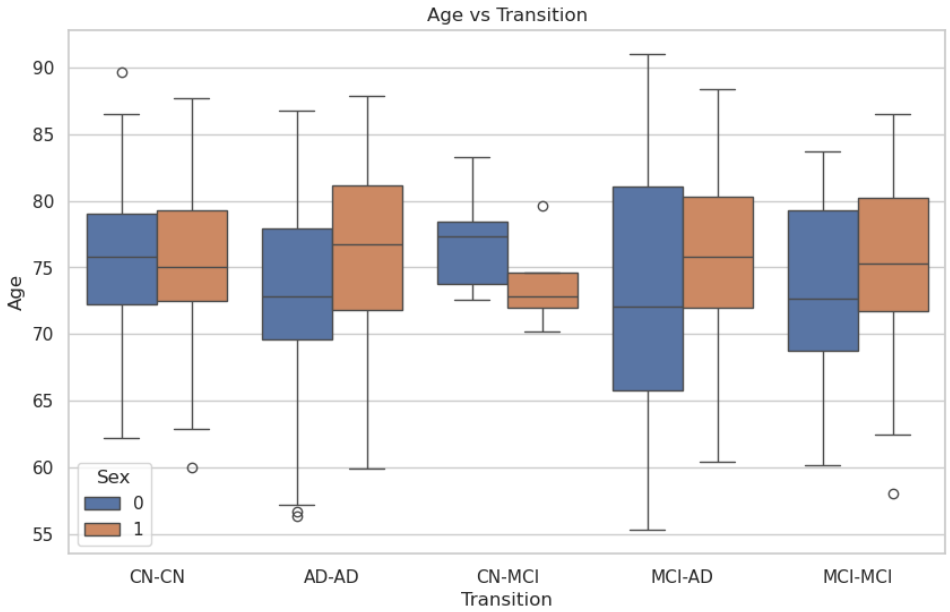

Figura 3.5: Comparing all three variables in a boxplot

4 Tratamento dos datasets

Após esta breve análise dos *datasets* procedemos ao tratamento dos mesmos. Este tratamento foi feito igualmente em todos os *datasets* de forma a estes estarem em termo de igualdade quando for para serem utilizados nos modelos. Este tratamento serviu para melhorar a consistência e qualidade dos dados que estavam inicialmente nos *datasets* para quando estes foram utilizados nos modelos de *machine learning*.

4.1 Remoção de colunas desnecessárias

Devido ao facto dos *datasets* terem um número exorbitante de colunas o nosso primeiro objetivo foi remover todas as colunas que sejam completamente desnecessárias para os modelos. A primeira coisa que fizemos foi verificar se existiam **missing values** em alguma coluna, pois isso tornaria essa coluna inconclusiva e se calhar seria necessária removê-la. Aconteceu que nestes *datasets* **não haviam missing values** de todo.

De seguida, fomos ver as **colunas com valores únicos**, ou seja, mesmo valor em todas as linhas. Neste caso encontramos uma diferença entre o *dataset* do hipocampo e o *dataset* do occipital. Enquanto que no *dataset* do occipital existem somente **147 colunas** com valores únicos, no caso do *dataset* do hipocampo já existem **159 colunas**. Fomos verificar e, exceto uma coluna do occipital, todas as outras também são colunas com valores únicos no *dataset* do hipocampo. Decidimos remover todas essas colunas nos dois *datasets*, as 147 no do occipital e as 159 no do hipocampo.

O próximo tratamento que fizemos vou descobrir **colunas que continham os mesmos valores**, ou seja, valores diferentes em todas as linhas, mas duas ou mais colunas continham esses valores iguais em todas as linhas. Também houve uma pequena discrepância entre o *dataset* do occipital e do hipocampo e, a certo ponto, reparamos que também dava resultados diferentes entre o de treino e o de teste do hipocampo. No occipital davam **115** colunas deste caso e no de treino do hipocampo **112** (no de teste também davam **115**). A forma como resolvemos isto foi a seguinte: no caso do occipital, como ele está sozinho, simplesmente se existiam 4 colunas com os mesmos valores, por exemplo, removíamos 3 e deixava-mos uma coluna com esses valores; no caso do hipocampo, resolvemos fazer o mesmo tratamento (o que fizemos no do occipital) no de treino e no de teste, removendo as mesmas colunas nos dois simultaneamente, pois como futuramente as previsões no *dataset* de teste vão ser feitas a partir do treino do *dataset* de treino este têm de ter **as mesmas colunas em comum**.

Para terminar em termos de remover colunas desnecessárias, resolvemos remover todas **as colunas do tipo object**, menos a **Transition** (visto que esta é a **target**). Esta decisão foi feita devido ao facto que é difícil encontrar padrões em *objects* e a maioria destes eram localização de ficheiros, versões de programas de diagnósticos, hash values ou coordenadas com valores muito similares.

4.2 Normalização dos dados

Em termos de normalização de dados, inicialmente ponderamos fazer **tratamento de outliers** e chegamos a testar alguns modelos com essas alterações, mas estes estavam a ficar *overfitted* por isso desistimos dessa ideia.

Tendo isto em conta, de forma a melhorar a transparência dos dados resolvemos normalizar os valores das colunas utilizando **MinMaxScaler** colocando todos os valores das colunas numéricas entre 0 e 1, pois existem vários modelos que necessitam dos dados normalizados para terem melhor performance, por exemplo, os de **ensemble learning**.

Depois de normalizarmos os dados, analisamos também a **correlação** de todas as *features* com a *Transition* onde verificamos que existiam várias variáveis com quase relevância nenhuma na previsão que íamos fazer, levando-nos a começar a pensar como faríamos a **Feature Selection**, pois é pouco provável um modelo conseguir fazer boas previsões estando a trabalhar simultaneamente com cerca de 1900 variáveis, o número que tinhamos atualmente depois de a remoção das desnecessárias. Depois desta visualização da correlação guardamos os novos *datasets* em novos '.csv' com o nome *treated* para termos uma distinção entre os *datasets* iniciais e os que já foram manuseados.

Antes da **Feature Selection**, a última coisa que fizemos foi passar todos os *float64* para *float32*, pois de acordo com as nossas pesquisas os modelos de *Machine Learning* funcionam melhor com variáveis de 32 bits do que com variáveis de 64 bits, devido ao facto que trabalham com menor carga.

4.3 Feature Selection e Feature Importance

Inicialmente, tentamos utilizar os três métodos principais dados nas aulas para análise e seleção de características num *dataset*: Mean Decrease Impurity (**MDI**), **Permutance Importance** e **SHAP** (SHapley Additive exPlanations). Apesar de serem excelentes técnicas para avaliar a importância das variáveis, no nosso caso os resultados obtidos apresentaram problemas, sendo elas uma redução a um número muito baixo de variáveis (cerca de 30/40) e um impacto negativo no desempenho nos modelos de classificação. Este comportamento pode ser atribuído a vários fatores como ruído nos dados ou a sensibilidade dos modelos às métricas de importância utilizadas.

Depois de muita pesquisa, deparamo-nos com uma abordagem alternativa que combina **RFECV (Recursive Feature Elimination with Cross-Validation)** e **PCA (Principal Component Analysis)** de forma a selecionar e reduzir as características mais relevantes:

- **RFECV (Recursive Feature Elimination with Cross-Validation)** - este método foi utilizado para selecionar iterativamente as características mais informativas com base em um modelo de classificação (nós utilizamos o **Random Forest Classifier**). O processo envolve a eliminação sequencial das variáveis menos importantes, avaliando a performance do modelo em cada etapa através de **Stratified Cross Validation**.
  - O número final de variáveis selecionadas foi **1650 no dataset do hipocampo** e de **1638 no dataset do occipital**
  - Foram eliminadas todas as variáveis irrelevantes ou redudantes. A evolução do desempenho do modelo em função do número de variáveis selecionadas foi monitorizado a partir de um gráfico, que demonstrou a relação entre o número de variáveis e a pontuação média do *cross validation*
- **PCA (Principal Component Analysis)** - após a seleção das variáveis pelo **RFECV**, aplicou-se o **PCA** para reduzir ainda mais a dimensionalidade do *dataset*, retendo **95% da variância explicada**. Este método transforma as variáveis selecionadas em um conjunto de componentes principais não correlacionados.
  - O PCA reduziu o número de componentes principais de **1650 para 90** no caso do hipocampo e de **1638 para 77** no caso do occipital. Um gráfico de barras foi gerado para visualizar a proporção da variância explicada por cada componente principal, indicando a contribuição de cada componente para o total da variância retida.

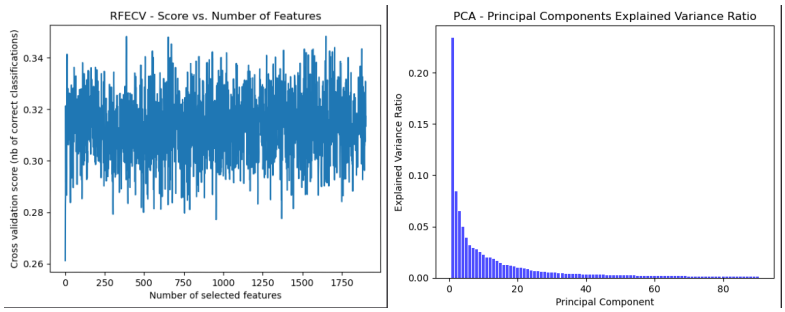

Figura 4.1: Gráficos do RFECV e do PCA

Esta abordagem combinada permitiu criar *dataset* finais para treino e teste com dimensões reduzidas e características otimizadas, garantindo um balanceamento adequado entre a complexidade do modelo e a preservação da informação nos dados. No final, os resultados obtidos com esta abordagem revelaram-se mais robustos e consistentes, evidenciado a eficácia da **combinação de RFECV e PCA** em comparação com os métodos de análise de importância que experimentamos inicialmente.

5 Modelos desenvolvidos

Depois de termos feito o tratamento dos dados, procedemos ao desenvolvimento dos modelos de Machine Learning. Para este projeto somente desenvolvemos algoritmos de aprendizagem supervisionada. Os modelos que desenvolvemos foram os seguintes: **Decision Tree, Random Forest, Support Vector Machine, Gradient Boost, XGBoost, Multi-Layer Perceptron , Stacking e Max Voting**. Todos estes modelos foram otimizados ao máximo e avaliados de forma a identificar a abordagem mais eficaz, considerando o desempenho em métricas como *accuracy*, *precision*, *F1-score* e *GridSearchCV*. Vale a pena mencionar que todos estes modelos utilizaram como *random_seed* o valor 2023.

5.1 Decision Tree Classifier

Este modelo organiza os dados na estrutura de uma árvore, fazendo decisões baseadas em divisões sequenciais de atributos. Cada nó representa uma condição e os seus ramos correspondem às possíveis saídas, enquanto as folhas representam os resultados finais. É intuitivo e eficaz para problemas simples, mas pode sofrer de *overfiting* em dados complexos como no nosso caso.

No *GridSearch* deste modelo testamos os seguintes hiperparâmetros: *max_depth, min_samples_split, min_samples_leaf, max_leaf_nodes, criterion, splitter, class_weight* e *ccp_alpha*. Depois de corrermos o *GridSearch* ficaram escolhidos os seguintes hiperparâmetros para o modelo final:

- **class_weight**: *balanced*
- **min_samples_leaf**: 3
- **max_leaf_nodes**: 25
- **splitter**: *random*

O melhor desempenho que este modelo alcançou no *score* público de **F1-macro** no **Kaggle** foi de **0.43799**, apesar de termos suspeitas de *overfitting*, pois no final nestes parâmetros só conseguimos uns míseros **0.1733**, tendo localmente uma *accuracy* de **0.34** para o hipocampo (o que é testado no *Kaggle*) e **0.25** no occipital. Os resultados serão mais explicitamente demonstrados em tabelas noutra parte do relatório.

5.2 Random Forest Classifier

Este modelo é um *ensemble* de várias árvores de decisão, construído de forma a aumentar a robustez e reduzir o *overfitting*. Cada árvore é treinada em uma amostra aleatória dos dados e o resultado final é obtido por votação. Isso melhora a generalização e é útil em cenários com alta dimensionalidade como o nosso.

No *GridSearch* deste modelo testamos os seguintes hiperparâmetros: *bootstrap, criterion, n_estimators, max_depth, min_samples_split, min_samples_leaf, max_features* e *class_weight*. Depois de correrrmos o *GridSearch* ficaram escolhidos os seguintes hiperparâmetros para o modelo final:

- **bootstrap**: *False*
- **class_weight**: *balanced*
- **max_depth**: 5
- **max_features**: log2
- **min_samples_split**: 10

O melhor desempenho que este modelo alcançou no *score* público de **F1-macro** no **Kaggle** foi de **0.31487**, estando sempre a rondar esse valor, apesar de localmente termos no final uma *accuracy* de **0.54** no hipocampo, sendo localmente um dos nossos melhores modelos neste *dataset*, e **0.26** no occipital.

5.3 Support Vector Machine

Este modelo está focado em encontrar um hiperplano que maximize a margem entre classes no espaço de características. Utiliza *kernel tricks* para lidar com dados não linearmente separáveis, oferecendo uma solução poderosa para problemas complexos de classificação como o nosso.

No *GridSearch* deste modelo testamos os seguintes hiperparâmetros: *C, gamma, kernel, degree, coef0, shrinking, probability, class_weight, decision_function_shape* e *tol*. Depois de correrrmos o *GridSearch* ficaram escolhidos os seguintes hiperparâmetros para o modelo final:

- **C**:1
- **class_weight**: *balanced*
- **decision_function_shape**: *ovo*
- **degree**: 2
- **gamma**: 0.01

O melhor desempenho que este modelo alcançou no *score* público de **F1-macro** no **Kaggle** foi de **0.47565**, sendo este o nosso melhor desempenho na competição, apesar de localmente termos tido apenas uma *accuracy* de **0.34** no hipocampo, fazendo crer que essa pontuação foi um acontecimento devido a *overfitting* no **Kaggle** neste caso, pois a nossa última submissão deste modelo deu somente **0.32869**. No occipital tivemos uma *accuracy* de **0.21**.

5.4 Gradient Boost Classifier

Este modelo é novamente um método de *ensemble*, sendo que este constrói modelos de forma iterativa, corrigindo os erros dos modelos anteriores. O aprendizado é conduzido pela minimização de resíduos, combinando modelos simples (como árvores de decisão) para obter um desempenho altamente preciso.

No *GridSearch* deste modelo testamos os seguintes hiperparâmetros: *n_estimators*, *learning_rate*, *max_depth*, *min_samples_split*, *min_samples_leaf*, *subsample*, *max_features* e *criterion*. Depois de correrrmos o *GridSearch* ficaram escolhidos os seguintes hiperparâmetros para o modelo final:

- **criterion**: *squared_error*
- **max_depth**: 5
- **max_features**: log2
- **min_samples_leaf**: 4

O melhor desempenho que este modelo alcançou no *score* público de **F1-macro** no **Kaggle** foi de **0.31428**, apesar de localmente termos no final uma *accuracy* de **0.43** no hipocampo e **0.31** no occipital.

5.5 XGBoost Classifier

Este modelo é uma versão otimizada do Gradient Boosting, projetada para ser altamente eficiente e escalável. Inclui técnicas como regularização para evitar *overfitting*, sendo amplamente usado em competições de *machine learning* por causa do seu desempenho superior. Uma pequena diferença é que neste modelo em concreto temos de utilizar a biblioteca **Label Encoder** para o mesmo funcionar.

No *GridSearch* deste modelo testamos os seguintes hiperparâmetros: *n_estimators*, *max_depth*, *learning_rate*, *subsample*, *colsample_bytree*, *gamma*, *reg_alpha*, *reg_lambda* e *min_child_weight*. Depois de correrrmos o *GridSearch* ficaram escolhidos os seguintes hiperparâmetros para o modelo final:

- **n_estimators**: 100
- **max_depth**: 3
- **learning_rate**: 0.1
- **subsample**: 0.8
- **colsample_bytree**: 0.8
- **gamma**: 0.2
- **reg_alpha**: 0
- **reg_lambda**: 0.1
- **min_child_weight**: 3

O melhor desempenho que este modelo alcançou no *score* público de **F1-macro** no **Kaggle** foi de **0.22349**, apesar de localmente termos no final uma *accuracy* de **0.51**, um valor bem mais alto, no hipocampo e **0.26** no occipital.

5.6 Multi-Layer Perceptron Classifier

Este modelo é um tipo de rede neural artificial que utiliza várias camadas de neurónios para capturar padrões complexos nos dados. Os pesos das conexões são ajustados iterativamente para minimizar uma função de perda, aprendendo representações não lineares.

No *GridSearch* deste modelo testamos os seguintes hiperparâmetros:*hidden_layer_sizes, activation, solver, alpha, learning_rate, max_iter, early_stopping, n_iter_no_change* e *batch_size*. Depois de correrrmos o *GridSearch* ficaram escolhidos os seguintes hiperparâmetros para o modelo final:

- **activation**: *tanh*
- **hidden_layer_sizes**: (100,50)
- **batch_size**: 32
- **max_iter**: 1000

O melhor desempenho que este modelo alcançou no *score* público de **F1-macro** no **Kaggle** foi de **0.31653**, apesar de localmente termos no final uma *accuracy* de **0.39**, um valor um bocado mais alto, no hipocampo e **0.20** no occipital, o mais baixo de todos.

5.7 Stacking Classifier

Este é outro modelo que utiliza o método de ensemble, sendo que, neste caso, este combina as previsões de múltiplos modelos base, utilizando um modelo de metanível (no nosso caso foi o Random Forest) para fazer a previsão final. Esta abordagem tira proveito das forças individuais dos modelos base, aumentando a precisão geral. Para este modelo utilizamos todos os modelos anteriormente mencionados, tendo testado diferentes configurações:

- **1ª configuração:** Dar stacking a 5 modelos de **Random Forest**, onde obtivemos uma *accuracy* local de **0.3770** e um F1-Score no Kaggle de **0.35492**.
- **2ª configuração:** Dar stacking a todos os modelos que desenvolvemos uma vez, onde obtivemos uma *accuracy* local de **0.3607** e um F1-Score no Kaggle de **0.35809**.
- **3ª configuração:** Dar stacking a 3 modelos de **Random Forest** e 2 de **XGBoost**, visto que são os modelos que estão a funcionar melhor locamente, onde obtivemos uma *accuracy* local de **0.3115** e um F1-Score no Kaggle de **0.35158**.

Conseguimos visualizar que o caso mais estável foi a configuração que utilizou os 6 modelos simultaneamente.

Utilizando estes casos para o occipital, ignorando as diferenças de *accuracy* de forma a manter o mesmo tratamento dos modelos, obtivemos os seguintes valores: **0.16** de *accuracy* no 1º caso, **0.25** de *accuracy* no 2º caso e **0.18** de *accuracy* no 3º caso.

5.8 Max Voting Classifier

O último modelo que desenvolvemos consiste em um outro método de *ensemble*, no qual múltiplos modelos base fazem previsões e a classe final é determinada pela maioria das "votações". Pesos podem ser atribuídos aos modelos para refletir sua relevância relativa, tornando-o uma solução simples e eficiente para melhorar a confiabilidade das previsões. Tal como no **Stacking Classifier** testamos variadas configurações:

- **1ª configuração**: calculamos a *accuracy* de cada modelo e se ela fosse, por exemplo, 54% davamos poder de voto de **5.4**. Obtivemos uma *accuracy* local de **0.5574**, o mais alto que obtivemos, e um F1-Score no Kaggle de **0.33413**.

- **2ª configuração**: novamente com a *accuracy*, se fosse entre 30-40 davamos poder de voto de 1, entre 40-50 davamos 2, acima de 50 davamos 3. Obtivemos uma *accuracy* local de **0.5574**, o mais alto que obtivemos novamente, e um F1-Score no Kaggle de **0.30085**.

- **3ª configuração**: na última configuração, fizemos como na primeira mas só usamos os modelos que tinham pode de voto acima de 4. Obtivemos uma *accuracy* local de **0.5082** e um F1-Score no Kaggle de **0.31031**.

Foi neste modelo que conseguimos chegar à nossa maior *accuracy* local, tornando assim, teoricamente, o nosso melhor modelo.

Utilizando estes casos para o occipital, ignorando as diferenças de *accuracy* de forma a manter o mesmo tratamento dos modelos, obtivemos os seguintes valores: **0.25** de *accuracy* no 1º caso, **0.26** de *accuracy* no 2º caso e **0.28** de *accuracy* no 3º caso. Isto demonstra ainda mais que o occipital está quase confirmado não ser um bom *dataset* para a previsão pois neste caso a *accuracy* subiu em sentido contrário do hipocampo algo inesperado.

6 Resultados e Análise Crítica

Depois de termos produzido todos os modelos e visto os seus resultados individualmente, resolvemos fazer umas tabelas para analisar melhor como cada modelo se comportava de forma a conseguirmos ver melhor as previsões para depois fazermos uma análise crítica. Todos estes resultados foram feitos de forma **local**, pois conseguimos ver mais dados sobre as previsões do que com os resultados do *Kaggle*.

6.1 Resultados localmente do DSHippo

Além dos valores de cada modelo em específico utilizando os hiperparâmetros definidos no *GridSearch* e especificados anteriormente no relatório, vamos também analisar as 3 configurações do **Stacking Classifier** e do **Max Voting Classifier**. Analisamos a *accuracy* de cada modelo, a *precision* em cada Transição em específico e o número de acertos no total e em cada transição.

Modelo	Accuracy	AD-AD(12)	CN-CN(19)	CN-MCI(2)	MCI-AD(14)	MCI-MCI(14)
DTC	0.34/21	0.33/7	0.54/13	0.00/0	0.10/1	0.00/0
RFC	0.54/33	0.50/8	0.53/16	0.00/0	0.33/2	0.78/7
SVM	0.34/21	0.44/8	0.47/7	0.00/0	0.10/1	0.31/5
GBC	0.43/26	0.20/2	0.52/14	0.00/0	0.38/3	0.44/7
XGBC	0.51/31	0.40/6	0.48/13	0.00/0	0.62/5	0.64/7
MLPC	0.39/24	0.31/5	0.57/8	0.00/0	0.25/3	0.44/8
STC1	0.38/23	0.44/7	0.58/14	0.00/0	0.14/2	0.00/0
STC2	0.36/22	0.30/6	0.55/12	0.00/0	0.33/3	0.25/1
STC3	0.31/19	0.29/4	0.52/12	0.00/0	0.19/3	0.00/0
MVC1	0.56/34	0.43/9	0.60/15	0.00/0	0.40/2	0.80/8
MVC2	0.56/34	0.45/9	0.58/15	0.00/0	0.60/3	0.70/7
MVC3	0.51/31	0.36/5	0.56/15	0.00/0	0.50/4	0.58/7

Tabela 6.1: Resultados obtidos a partir do Classification Report de cada modelo no *dataset* hipocampo de treino

6.2 Resultados localmente do DSOcc

Depois de termos feito a tabela para os resultados do hipocampo, procedemos a formular esta tabela para o *dataset* do occipital para confirmarmos a nossa hipótese inicial que o *dataset* do hipocampo seria sempre melhor para fazer previsões. Analisamos novamente os mesmos modelos, configurações e valores.

Modelo	Accuracy	AD-AD(12)	CN-CN(19)	CN-MCI(2)	MCI-AD(14)	MCI-MCI(14)
DTC	0.25/15	0.25/1	0.31/8	0.12/1	0.11/1	0.29/4
RFC	0.26/16	0.29/4	0.37/7	0.00/0	0.20/3	0.17/2
SVM	0.21/13	0.32/6	0.24/4	0.00/0	0.10/1	0.17/2
GBC	0.31/19	0.25/3	0.31/10	0.00/0	0.33/2	0.36/4
XGBC	0.26/16	0.40/4	0.26/7	0.00/0	0.22/2	0.20/3
MLPC	0.20/12	0.17/3	0.25/4	0.00/0	0.12/2	0.30/3
STC1	0.16/10	0.18/5	0.40/2	0.12/1	0.09/1	0.11/1
STC2	0.25/15	0.27/6	0.31/4	0.00/0	0.09/1	0.27/4
STC3	0.18/11	0.21/6	0.25/2	0.20/1	0.18/2	0.00/0
MVC1	0.25/15	0.29/4	0.33/9	0.00/0	0.09/1	0.11/1
MVC2	0.26/16	0.27/4	0.33/9	0.00/0	0.12/1	0.20/2
MVC3	0.28/17	0.30/3	0.36/10	0.00/0	0.20/2	0.17/2

Tabela 6.2: Resultados obtidos a partir do Classification Report de cada modelo no *dataset* occipital

6.3 Análise Crítica

A primeira observação que conseguimos fazer foi que ao adicionarmos no tratamento de dados a utilização do **RFECV e PCA** na escolha das variáveis a utilizar aumentou muito a qualidade dos dados, tornando-os mais claros e fáceis de utilizar pelos *datasets*, tendo um tempo de execução menor.

A utilização de **GridSearch** de forma extensa, podendo às vezes demorar quase uma hora para executar, tornou muito fácil a escolha dos hiperparâmetros utilizados nos modelos, demonstrando a sua utilidade quando se trabalha com modelos de Machine Learning.

Termos utilizado vários modelos diferentes deixou-nos a entender melhor quais eram melhores nas previsões e quais eram piores, tendo uma melhor noção se era um problema nos dados ou no facto de que o modelo que estávamos a utilizar era mais fraco em comparação ao outro.

A utilização do Kaggle no entendimento dos modelos ajudou-nos a entender de alguma forma se os nossos modelos realmente eram muito bons ou se encontravam com *overfit*. Por exemplo, o facto do **SVM** ter dado **0.47**, um valor bem elevado no Kaggle, mas somente **0.34** localmente deixou-nos com a impressão que havia algo de errado e fez-nos fazer algumas alterações nos

modelos. Tal como aconteceu com o **Max Voting Classifier**, que localmente tinha uma *accuracy* de **0.56** mas no Kaggle nunca chegou a ultrapassar os **0.33** de F1-Score.

Depois de analisarmos todos os modelos em ambos os *datasets*, conseguimos confirmar com clareza a nossa hipótese inicial de que o *dataset* do occipital não foi feito para a previsão da progressão de défice cognitivo leve (MCI) para a Doença de Alzheimer (AD), nem para qualquer outra transição presente no *dataset*. Em termos de *accuracy* geral, reparamos logo que no *dataset* do occipital o máximo foi de **0.31**, sendo que foi o caso em que acertou 10 CN-CN, a única transição que até teve algumas boas previsões. Já no caso do *dataset* do hipocampo, conseguimos chegar a uma *accuracy* de **0.56**, o que não é um valor muito elevado, mas ainda aceitável dentro das circunstâncias. Além disso, todas as transições, no caso do occipital, tiveram em todos os modelos *precision* muito baixa, incluindo a **Transição MCI-AD** a mais importante de prever nesta experiência.

Falando de cada transição em específico no caso do hipocampo, o *dataset* que, depois das observações, é claramente o ideal para previsões, temos alguns detalhes para refletir:

- **AD-AD**: Apesar de ter uma boa *accuracy* (acerta grande parte deles) na maioria dos casos(0.75 num dos casos), a sua *precision* deixa a desejar(0.50 no máximo), ou seja, os modelos acham que vários pacientes também se encontram nessa transição apesar de não estarem.

- **CN-CN**: esta deve ser a Transição que tem melhor *accuracy* e melhor *precision*, mas isso também acontece devido ao facto de ser a Transição que existe mais no *dataset* logo é a mais treinada.

- **CN-MCI**: esta transição foi impossível de ser prevista devido ao facto de só existirem 10 casos dela no *dataset* e 2 para serem previstos no teste. Acontece que na maioria dos casos, os modelos nunca previam que seria esta transição, logo também não estragou muito as métricas.

- **MCI-AD**: a transição mais importante do *dataset* e que no final das contas foi das mais complicadas de se prever. Exceto no caso do XGBoost Classifier e dos Max Voting Classifier que ainda tiveram uma boa *precision*, mas mesmo assim ainda uma *accuracy* um bocado baixa.

- **MCI-MCI**: esta transição teve os resultados provavelmente mais constantes, exceto o facto que ou tinha *precision* e *accuracy* boas, ou terríveis.

Em suma, o *dataset* do occipital claramente não foi feito para este tipo de previsões com qualquer dos modelos que nós formulamos e o *dataset* do hipocampo com o tratamento e os modelos certos consegue de alguma forma fazer as previsões pedidas, apesar de ainda com alguma dificuldade, pois um *dataset* destes é muito complexo para estes modelos básicos por mais treinados que sejam.

7 Conclusão

Este trabalho prático explorou a concepção e otimização de modelos de Machine Learning para prever a progressão de MCI para AD usando características radiômicas de imagens de ressonância magnética do hipocampo e do lobo occipital. A análise comparativa dos *datasets* confirmou que o hipocampo é mais relevante para este tipo de previsão, enquanto o *dataset* do occipital apresentou baixo desempenho consistentemente.

As principais conclusões foram as seguintes: **Qualidade dos Dados e Relevância Regional** - O *dataset* do hipocampo demonstrou maior potencial preditivo, com um desempenho superior em *accuracy* e *precision* para a maioria das transições analisadas. O *dataset* do occipital mostrou-se inadequado para prever a progressão do MCI para AD, apresentando baixos valores em todas as métricas de desempenho e transições; **Impacto do Pré-Processamento** - Técnicas como RFECV e PCA foram cruciais para reduzir a dimensionalidade e melhorar a qualidade dos dados. A normalização dos dados e a remoção de colunas redundantes ou irrelevantes aumentaram também a eficiência dos modelos; **Desempenho dos Modelos** - entre os modelos desenvolvidos, o SVM apresentou o melhor desempenho no *Kaggle*, mas evidências de overfitting limitaram sua confiabilidade. Os métodos ensemble, como Max Voting Classifier e Stacking, forneceram resultados estáveis, com o Max Voting alcançando a maior *accuracy* localmente (0.56 no hipocampo). A transição CN-CN foi a mais fácil de prever devido à sua maior representatividade no *dataset*, enquanto CN-MCI e MCI-AD apresentaram desafios significativos, particularmente devido à escassez de casos para treino.

As principais recomendações são as seguintes: **Aumentar a Base de Dados** - Priorizar a coleta de mais pacientes, especialmente nas transições menos representadas como CN-MCI e MCI-AD, para melhorar a robustez dos modelos. Expandir o *dataset* do hipocampo com mais amostras para consolidar sua eficácia preditiva; **Explorar Modelos Avançados** - Considerar a aplicação de redes neurais profundas ou métodos híbridos que combinem abordagens supervisionadas e não supervisionadas para lidar melhor com a complexidade do problema;**Cross Validation Mais Robusta** - Implementar *stratified cross validation* em todas as fases do treinamento para reduzir o risco de *overfitting* e garantir maior generalização; **Incluir Variáveis Contextuais** - Adicionar dados clínicos e históricos, além de características radiômicas, para enriquecer as informações disponíveis aos modelos.

Em suma, o trabalho demonstra que, apesar das limitações dos *datasets* e modelos básicos utilizados, é possível alcançar resultados promissores. Com investimentos adicionais em dados, modelagem e técnicas de análise, há grande potencial para avanços significativos na previsão da progressão de MCI para AD.