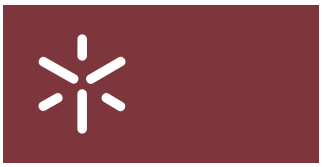
# Dados e Aprendizagem Automática

Trabalho em Grupo


| **Filipe Pereira** | **João Lopes** | **João Vale** | **Luís Barros** |
| --- | --- | --- | --- |
| pg55941 | pg55964 | pg55951 | pg55978 |

Grupo 45



Universidade do Minho

Ano Letivo 2024/2025

# Índice

# Figuras

# 1 Introdução

O presente relatório apresenta o desenvolvimento e análise de modelos de **Machine Learning** aplicados à previsão da progressão de Comprometimento Cognitivo Ligeiro (MCI) para a doença de Alzheimer (AD), utilizando dados do Alzheimer's Disease Neuroimaging Initiative (ADNI). Através de dados **radiômicos** obtidos de ressonâncias magnéticas, exploramos a relação entre mudanças estruturais no cérebro e a evolução clínica da doença.

Este projeto visa contribuir para o diagnóstico precoce da AD, possibilitando intervenções mais eficazes no combate à progressão da doença.

# 2 Áreas abordadas e Objetivos

## 2.1 Áreas abordadas

Este trabalho foca-se na aplicação de técnicas de **Machine Learning** para o diagnóstico precoce de doenças neurodegenerativas. Desta forma, o estudo explora a **relevância de regiões específicas** do cérebro (hipocampo e lobo occipital) através da análise de características **radiômicas** extraídas de ressonâncias magnéticas.

Adicionalmente são abordadas metodologias para a **exploração** e **processamento** destes dados médicos, assim como o **desenvolvimento**, **treino** e **otimização** dos modelos preditivos e a **comparação** dos resultados obtidos com diferentes abordagens.

## 2.2 Objetivos

Os principais objetivos deste projeto passam por:

- **Analisar a relevância do hipocampo para a progressão de MCI para AD**: ou seja, avaliar se as características desta região são mais informativas que as do lobo occipital, uma área menos associada a esta doença.
- **Desenvolver modelos ML eficazes**: criar, treinar e otimizar modelos de Machine Learning para prever a progressão da doença com base no conjunto de dados do hipocampo.
- **Criticar e validar os resultados obtidos**: garantir que os modelos desenvolvidos são uteis para o contexto e identificar as suas limitações.

## 2.3 Abordagem proposta

De modo a completar os objetivos estabelecidos, o grupo achou que seria adequado fazer uma boa **exploração e preparação** dos dados disponíveis, preparar e ajustar **modelos** adequados ao tipo de problema, sendo este **categórico** com a presença de **cinco classes** distintas, e ainda **validar** os resultados tendo em conta a métrica **F1-macro**, uma métrica capaz de avaliar o desempenho considerando todas as classes igualmente.

# 3 Metodologia

A metodologia adotada para este trabalho segue o padrão **CRISP-DM**, devido à sua abordagem iterativa e abrangente, permitindo estruturar o projeto desde a compreensão inicial do problema até a avaliação dos resultados.

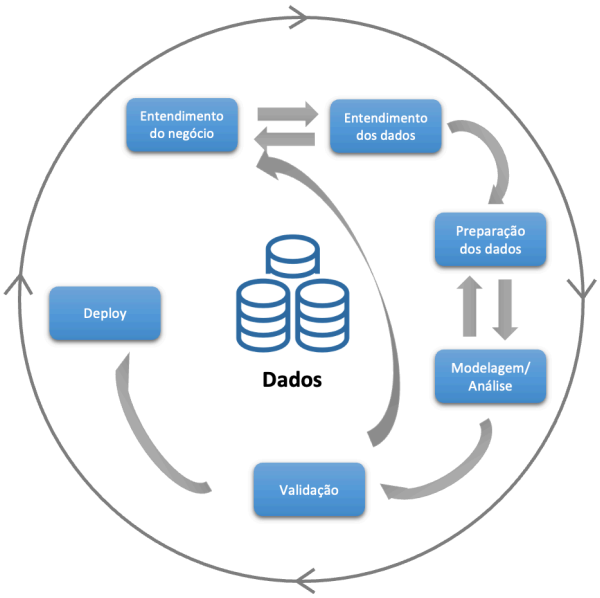



Figure 1: Estratégia CRISP-DM

## 3.1 Entendimento do negócio

Como já mencionado neste documento, o problema em questão é a possível progressão de um estado de doença ligeiro, MCI, para um estado mais grave, AD. Tendo este entendimento em mãos, torna-se clara a necessidade clínica de prever o quanto antes se essa progressão irá acontecer. Para isso, o uso de modelos de machine learning torna-se crucial na medida em que permite intervenções precoces por parte médica.

## 3.2 Entendimento dos dados

Numa análise mais superficial aos dados fornecidos pelo **ADNI**, percebemos que se tratavam de dados radiômicos do hipocampo e do lobo occipital. Tratando-se de análises clínicas de 305 pacientes, com 5 diferentes transições associadas, havendo dentro destes 173 homens e 132 mulheres com uma distribuição de idades entre os 55 e os 91 anos.

## 3.3 Preparação dos dados

De modo a tornar os dados mais **limpos** e **claros** para os nossos modelos começamos por remover features que **não** apresentassem informação relevante, como por exemplo as versões das ferramentas utilizadas. **Transformámos** também colunas não numéricas em numéricas de modo a extrair alguma informação que pudesse ser relevante para o problema e **escalámos** os dados para o uso de modelos sensíveis à magnitude das features, como o SVM ou o XGBoost.

## 3.4 Modelagem

No que toca a modelos de ML, começamos por abranger diversos adequados a problemas categóricos e fomos restringindo conforme a sua performance e resultados obtidos. Seguem-se os modelos testados:

| Modelo | Resultados | Performance/Speed | Tipo |
|---|---|---|---|
| **Logistic Regression** | Moderados | Rápido | Regressão Linear |
| **RandomForest** | Moderados | Rápido | Árvores de decisão |
| **XGBoost** | Bons | Rápido | Gradiente |
| **GradientBoost** | Bons | Médio | Gradiente |
| **CatBoost** | Bons | Muito Lento | Gradiente |
| **LightGBM** | Bons | Rápido | Gradiente |
| **SVM** | Moderados | Rápido | Métodos geométricos |
| **Redes Neurais** | Bons | Lento | Redes de neurônios |

No entanto, o modelo final trata-se de um **Ensemble Stacking** de **Random Forest**, **SVM**, **Logistic Regression**, **XGBoost** e **LightGBM**, principalmente por se tratarem de modelos com diferentes abordagens e portanto capazes de capturar diferentes padrões. Será abordado em detalhe no decorrer do relatório.

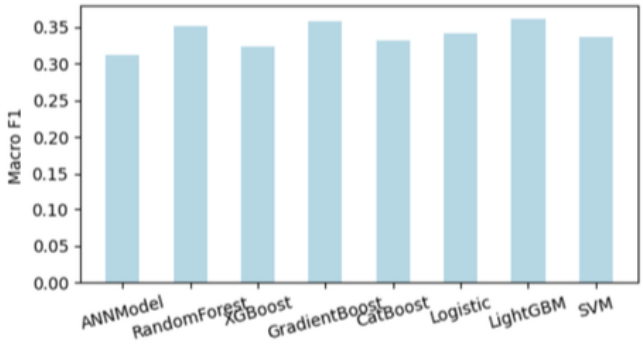



Figure 2: Comparação de resultados com todas as features.

## 3.5 Validação

Devido ao problema de **desbalanceamento** das classes, a utilização de **cross-validation** com uma divisão **estratificada** dos dados foi fundamental. Esta abordagem assegura que cada conjunto de treino e teste tenha uma **distribuição balanceada** das classes, o que é crucial para a avaliação correta dos modelos em problemas multiclasse.

Para evitar **overfitting**, foi reservado um conjunto de dados **exclusivo** para testes finais, garantindo que os modelos não tivessem acesso a esse conjunto durante o treinamento e pudessem **generalizar** corretamente.

Para avaliar o desempenho dos modelos, utilizamos a métrica **F1-macro**, que é apropriada para problemas multiclasse **desbalanceados**, além de outras métricas como **ROC** e **AUC**. A **matriz de confusão** foi também empregada para examinar o desempenho dos modelos em cada classe individualmente.

# 4 Exploração de Dados

Embora admirada por poucos, exploração de dados é uma etapa crucial no treinamento de modelos de Machine Learning. Entender os dados com que estámos a lidar é fundamental para um conhecimento maior na hora de escolher e otimizar os modelos.

## 4.1 Exploração Inicial

Assim, começamos por analisar a distribuição da variável target e observamos um grande desbalanceamento, como demonstra a seguinte figura, o que nos indica que devemos estar atentos e ter este detalhe em consideração futuramente.

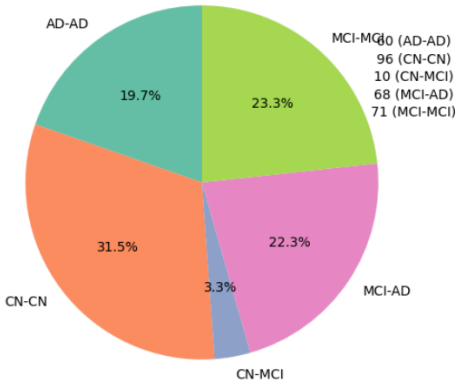



Figure 3: Distribuição de classes.

e analisamos também a distribuição de idades no nosso dataset.

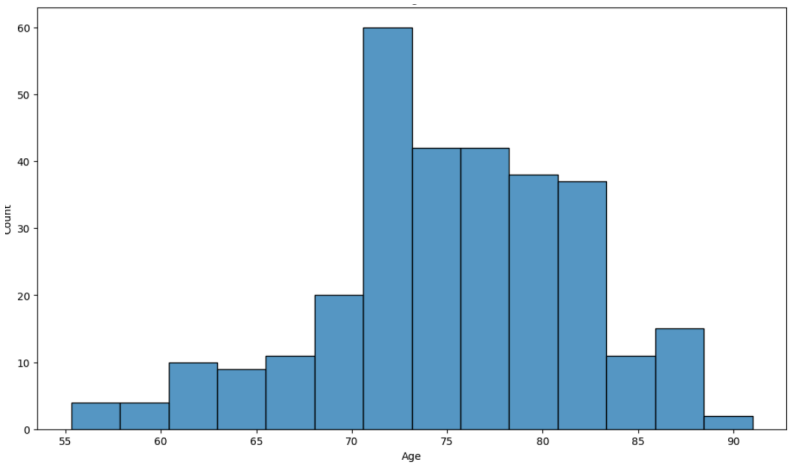



Figure 4: Distribuição de idades.

Seguidamente, tentamos encontrar alguma relação entre o **sexo** e a **idade** com a variável target, no entanto, numa análise superficial, não nos pareceu existir alguma relação forte (figura 5). Caso que se veio a confirmar na **matriz de correlação** entre estas três variáveis (figura 6).

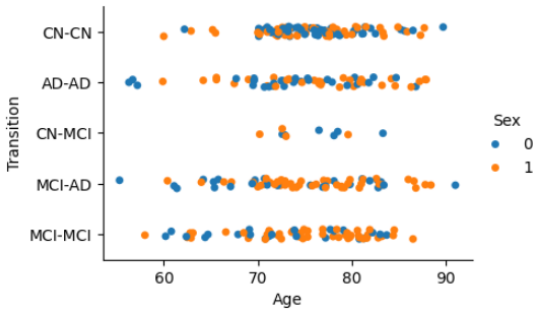



Figure 5: Distribuição de sexo e idade com o target.

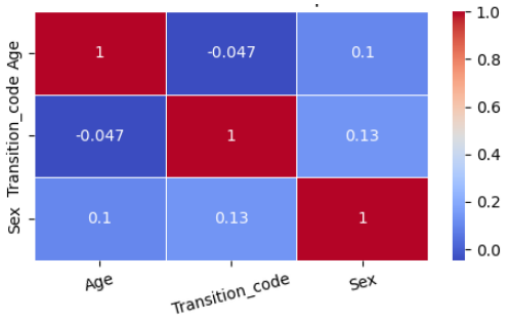



Figure 6: Correlação entre sexo, idade e target.

## 4.2 Features não Numéricas

Foram identificadas 20 features não numéricas nos datasets disponibilizados. Entre elas, analisámos quais poderiam fornecer informações relevantes para o problema. Concluímos que apenas *diagnostics_Mask-original_BoundingBox* e *diagnostics_Mask-original_CenterOfMass* seriam adequadas para feature engineering, de modo a extrair dados úteis para os modelos.

As restantes features foram descartadas.

## 4.3 Correlação

### 4.3.1 Variáveis Independentes e Target

Seguimos para a avaliação da relação entre as features e a variável target e concluímos que de facto haviam variáveis com relação **extremamente baixa** com o target e que, dentro das variáveis **mais** relacionadas, ainda assim a relação era **menor** do que **esperávamos**.

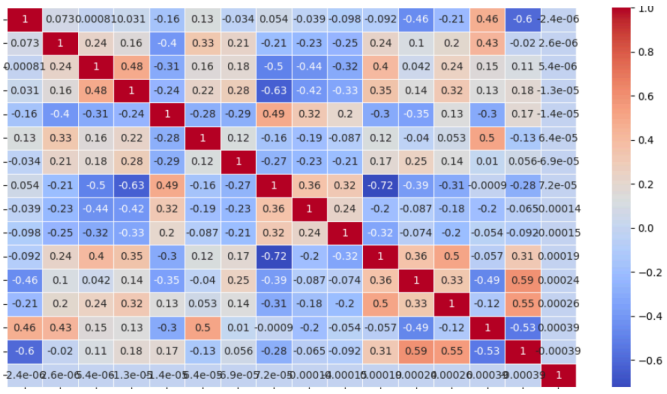


Figure 7: Features com menor correlação com o target.

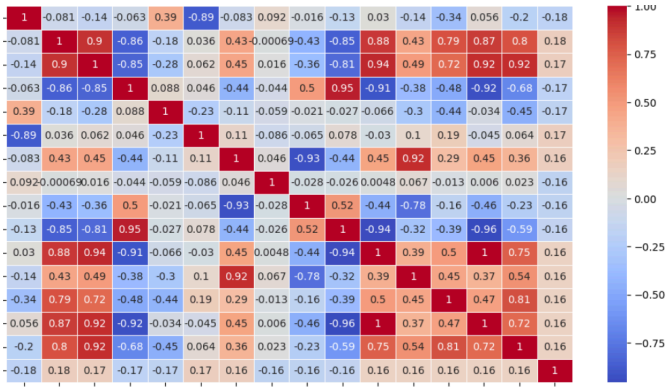


Figure 8: Features com maior correlação com o target.

Como observado na última linha de cada tabela (linha de correlação com o target), a figura 7 apresenta valores extremamente infímos e, numa primeira análise, irrelevantes. Já na figura seguinte, figura 8, é possível observar valores mais notáveis, acima de **0.1**.

### 4.3.2 Variáveis Independentes Entre Si

Dentro das variáveis independentes também consideramos útil analisar a sua correlação entre si, levando a concluir que de facto existe uma boa proporção de features que está **altamente** correlacionada, sendo um indicativo de que eventualmente seria boa ideia descartá-las.

Segue um exemplo de matriz de correlação que demonstra a existência de diversas features com uma correlação de 100%, apresentando valores como **1** ou **−1** (excluindo a diagonal).

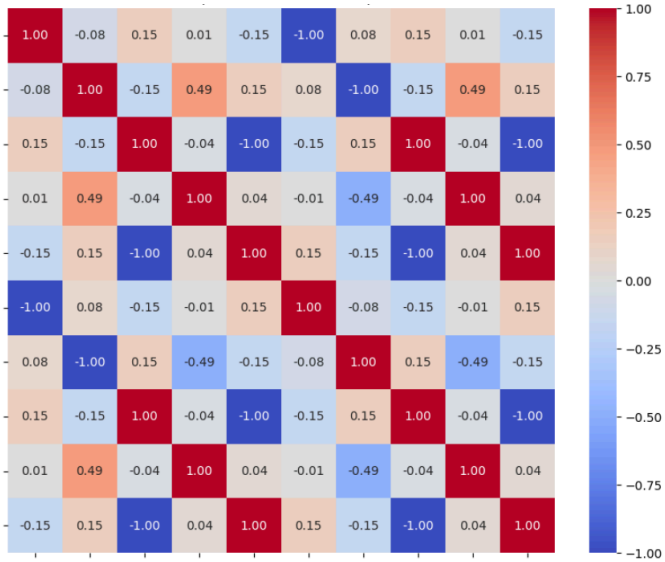


Figure 9: Features com muita correlação entre si.

Em contrapartida, fomos capazes de observar também que existem features pouco relacionadas entre si e que podem trazer mais valias para o nosso modelo, como segue o exemplo:

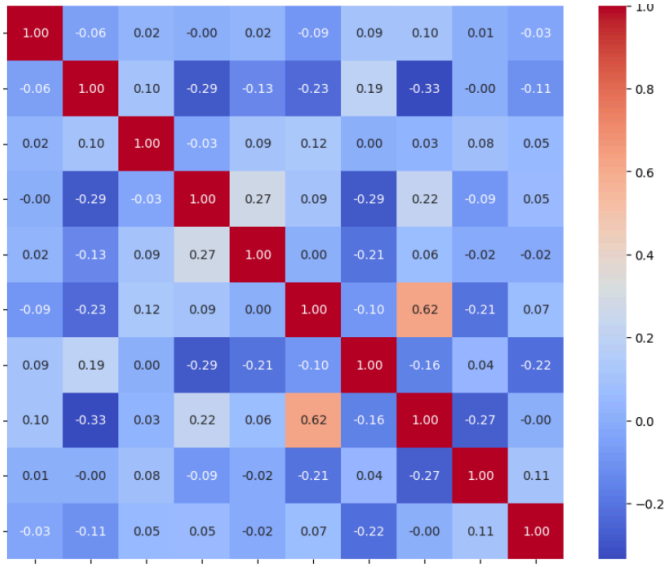



Figure 10: Features com pouca correlação entre si.

Concluímos com esta exploração que existem diversas vertentes e detalhes a ter em atenção numa fase posterior, levando a possíveis melhores resultados nos nossos modelos.

Estes processos de exploração foram aplicados ao dataset do hipocampo e **replicados** no dataset de controlo relativo ao lobo occipital, resultando em **conclusões semelhantes**.

# 5 Processamento e Preparação de Dados

O processamento de dados desempenha um papel crucial na eliminação ou minimização de **ruído**, tornando o dataset mais **limpo** e adequado para os modelos de Machine Learning.

Nos datasets analisados, não foram identificados **missing values** ou **outliers** significativos que justificassem tratamento ou remoção. Contudo, a escolha **criteriosa de features** merece destaque. Com 2162 colunas e apenas 305 entradas, é essencial realizar uma **seleção** cuidadosa das features mais relevantes, garantindo a eficácia e eficiência dos modelos.

## 5.1 Processamento de features não numéricas

Como já mencionado, começamos por aplicar **feature engineering** às features *diagnostics_Mask-original_BoundingBox* e *diagnostics_Mask-original_CenterOfMass*, trasnformando estas listas de dados em features individuais, como segue o exemplo:

| **diagnostics_Mask-original_BoundingBox** |
|---|
| (103, 113, 93, 36, 30, 71) |
| (32, 104, 3, 54, 10, 89) |

transforma-se em:

| **x_min** | **y_min** | **largura** | **altura** | **profundidade** | **extra** |
|---|---|---|---|---|---|
| 103 | 113 | 93 | 36 | 30 | 71 |
| 32 | 104 | 3 | 54 | 10 | 89 |

## 5.2 Normalização e Padronização dos Dados

A utilização de **data scalers** é essencial em Machine Learning, especialmente quando os modelos dependem de métricas baseadas em distância, como regressões ou algoritmos baseados em gradiente. **Escalar** os dados garante que todas as features sejam tratadas na mesma escala, evitando que variáveis com valores maiores dominem o treinamento.

## 5.3 Feature Selection

### 5.3.1 Abordagem Inicial

Durante a **exploração de dados**, identificamos que features com **baixa** correlação com o target e **alta** correlação entre si podem ser **candidatas** à remoção. Aplicando a abordagem **iterativa** da estratégia CRISP-DM, determinamos os seguintes thresholds: features com correlação com o target inferior a *0,005* foram descartadas, enquanto, para pares de features com correlação superior a *0,9* entre si, uma delas foi removida.

Obtivemos assim um novo dataframe composto por **655** features!

### 5.3.2 Abordagem mais profunda (SHAP Values)

Apesar de já termos reduzido significativamente o número de features, de 2162 para 655, identificamos que essa quantidade ainda poderia levar a **overfitting**, considerando que o dataset contém apenas 305 entradas. Esse **desequilíbrio** entre o número de features e as amostras dificulta a **generalização** dos modelos e aumenta a **complexidade** e o custo computacional.

Numa tentativa de melhorar os resultados, começamos por analisar as **feature importances**, mas os resultados não foram satisfatórios. Foi então que consideramos o uso de SHAP Values, que oferecem uma análise mais **detalhada** sobre a **contribuição** de cada feature para as previsões dos modelos. Essa abordagem mostrou-se promissora por capturar interações e impactos das features, fornecendo uma base mais **robusta** para a seleção.

Após aplicarmos novamente uma abordagem iterativa, identificamos o **XGBoost** como o modelo que gerava os SHAP Values mais informativos. Isso permitiu uma redução adicional de features, de **655** para **91**, **equilibrando** a capacidade de generalização dos modelos com a necessidade de **evitar overfitting** e mantendo um custo computacional mais **eficiente**.

## 5.4 SHAP Values

Como mencionado anteriormente, esta ferramente foi encontrada como a mais adequada para o problema da seleção de features. Apresentamos abaixo a estratégia detalhada que nos permitiu reduzir o número de features de 655 para 91.

### 5.4.1 Modelos Utilizados

Iniciamos o processo utilizando *Bayesian Optimization* para otimizar os *hiperparâmetros* de diversos modelos: XGBoost, Random Forest, Logistic Regression, LightGBM e SVM. Todos foram treinados com o conjunto inicial de **655** features.

Após otimizar os modelos, obtivemos os SHAP Values detalhados de cada um, avaliando o impacto de cada feature em suas previsões. Contudo, a análise indicou que o **XGBoost** se destacava como o modelo mais informativo para essa tarefa.

### 5.4.2 Exploração dos SHAP Values

Encontramos três abordagens possíveis para a análise de SHAP Values:

- SHAP Values **locais**: esta abordagem é muito exaustiva por se tratar de um dataset com imensas features e entradas, no entanto é recompensada pela sua precisão, pois traz informação detalhada de cada feature por entrada individualmente.
- SHAP Values **globais**: contrária à abordagem, esta é muito abrangente por se tratar da média de SHAP Values por feature. No entanto torna-se vantajosa por permitir a remoção rápida de todas as features que apresentem uma média de 0.
- SHAP Values por **classe**: sendo a abordagem mais equilibrada, procura analisar a média dos SHAP values por cada classe, trazendo consigo os aspetos positivos e negativos das anteriores.

Dentre as três abordagens, optamos por explorar os SHAP Values por Classe, pois esta estratégia ofereceu um bom desempenho entre granularidade e eficiência computacional.

Para cada classe, utilizamos critérios definidos iterativamente para identificar features candidatas à remoção. Segue um exemplo prático para a classe 0 (AD-AD):

| | importance_mean | importance_max | importance_std | positive_ratio | negative_ratio | skewness |
|---|---|---|---|---|---|---|
| **count** | 655.000000 | 655.000000 | 655.000000 | 655.000000 | 655.000000 | 146.000000 |
| **mean** | 0.008114 | 0.035725 | 0.006544 | 0.098692 | 0.122519 | 2.888779 |
| **std** | 0.030074 | 0.113440 | 0.020675 | 0.220326 | 0.259070 | 1.980686 |
| **min** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | -0.312767 |
| **25%** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 1.309063 |
| **50%** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 2.514550 |
| **75%** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 4.107987 |
| **max** | 0.307048 | 1.250126 | 0.211041 | 0.979508 | 0.979508 | 7.713615 |

```
discard_features_0 = shap_importances_xgb_0[
            (shap_importances_xgb_0["importance_mean"] <= 0.0) |
            (shap_importances_xgb_0["importance_std"] <= 0.01) |
            (shap_importances_xgb_0["skewness"].abs() > 1) |
            (shap_importances_xgb_0["negative_ratio"] - shap_importances_xgb_0["positive_ratio"] > 0.7)
        ]["feature"].tolist()
```

Figure 11: Seleção de features com SHAP Values.

- Features com média de SHAP Values igual a 0.
- Desvio padrão dos SHAP Values inferior a 0.01.
- Assimetria dos valores (skewness) superior a 1.
- Discrepância entre valores positivos e negativos superior a 0.7.

Após identificar os candidatos para cada classe, adotamos uma estratégia de interseção. Todas as features que foram classificadas como não importantes em pelo menos 4 das 5 classes foram removidas. Esta abordagem garantiu que as features retidas fossem relevantes para a maioria dos cenários analisados, reduzindo ainda mais o risco de overfitting e mantendo a capacidade de generalização dos modelos.

# 6 Modelagem e Treino

Para alcançar os melhores resultados, selecionamos modelos robustos e implementamos métodos de otimização de hiperparâmetros.

## 6.1 Estratégia de Otimização

Optámos por usar o método *Grid Search* combinado com *cross-validation* estratificado e balanceado, o que garantiu uma avaliação robusta e mais justa do desempenho dos modelos.

## 6.2 Modelos Utilizados e Hiperarâmetros Otimizados

### 6.2.1 RandomForest

Modelo baseado num ensemble de árvores de decisão que usa amostras aleatórias e combina os resultados para melhorar a precisão e reduzir overfitting.

Procuramos com auxilio do grid search, melhorar os seguintes hiperparametros:

- **nr_estimators**: Número de árvores no ensemble.
- **max_depth**: Profundidade máxima das árvores.
- **min_samples_split**: Mínimo de amostras para dividir um nó.
- **min_samples_leaf**: Mínimo de amostras em uma folha.
- **bootstrap**: Impacta a diversidade das árvores.
- **max_features**: Número máximo de features consideradas em cada divisão.
- **class_weigth**: Ponderação das classes para lidar com desbalanceamento.

### 6.2.2 XGBoost

Modelo de boosting que adiciona árvores de maneira iterativa, corrigindo os erros das anteriores. Este ensemble é normalmente bom para capturar relações complexas.

Procuramos com auxilio do grid search, melhorar os seguintes hiperparametros:

- **nr_estimators**: Número de árvores no ensemble.
- **max_depth**: Profundidade máxima das árvores.
- **learning_rate**: Taxa de aprendizagem.
- **subsample**: Proporção de amostras usadas por árvore.
- **colsample_bytree**: Proporção de features usadas por árvore.
- **min_child_weight**: Número mínimo de amostras em um nó folha.
- **objective**: Define o tipo de problema. Para multiclasse, utilizamos 'multi:softprob' pois gera probabilidades.

### 6.2.3 LightGBM

Modelo de boosting baseado em histogramas, eficiente para grandes datasets.

Procuramos com auxilio do grid search, melhorar os seguintes hiperparametros:

- **nr_estimator**: Número de árvores.
- **max_depth**: Profundidade máxima das árvores.
- **num_leaves**: Número máximo de folhas por árvore.
- **subsample**: Proporção de amostras usadas por árvore.
- **colsample_bytree**: Proporção de features usadas.
- **class_weigth**: Ponderação das classes para lidar com desbalanceamento.
- **objective**: Define o tipo de problema.

### 6.2.4 SVM

Modelo que encontra hiperplanos para separar classes com margens máximas. Usa kernels para encontrar relações complexas.

Procuramos com auxilio do grid search, melhorar os seguintes hiperparametros:
- **C**: Penalidade por erros de classificação.
- **kernel**: Função usada para calcular similaridades.
- **gamma**: Determina a influência de cada amostra no kernel.
- **degree**: Para multiclasse, define o método de decisão (ovo para um-contra-um).
- **class_weigth**: 'balanced'

#### 6.2.5 LogisticRegression

Modelo linear que usa a função sigmoide para prever probabilidades. Simples e eficiente em problemas lineares e portanto escolhido por nós para manter um equilibrio durante o ensemble stacking.

Procuramos com auxilio do grid search, melhorar os seguintes hiperparametros:
- **C**: Inverso da regularização.
- **penalty**: Tipo de regularização.
- **max_iter**: Máximo de iterações para convergência.
- **class_weigth**: 'balanced'

#### 6.2.6 Redes Neurais

Redes neurais são inspiradas no funcionamento do cérebro humano e são capazes de aprender padrões complexos. No entanto **não** foi escolhida para modelo final por não nos inspirar confiança nos resultados e na sua **generalização**. Para além disso, por se tratar de um modelo **complexo**, o facto de apenas termos 305 entradas pode levar a que o modelo encontre demasiado bem os padrões e ocorra um overfitting maior do que em modelos mais simples.

Ainda assim, consideramos melhorar os seguintes hiperparâmetros:
- **Numero de camadas e neurónios**: controla a capacidade de aprendizagem.
- **Taxa de aprendizagem**: define a velocidade de atualização dos pesos.
- **Funções de ativação**: definem como as redes aprendem relações não lineraes.

#### 6.2.7 Ensemble Stacking

Pensamos na utilização de RandomForest, XGBoost, LightGBM, SVM e LogisticRegression juntos num stacking ensemble, por possuirem caracteristicas **complementares**, aumentando a probabilidade de ter um bom **desempenho geral**.

Vantagens deste stacking:
- **Diversidade**: por apresentarem uma boa diversidade de algoritmos, são capazes de combinar os pontes fortes de cada um, encontrando diferentes padrões e melhorando a robustez.
- **Redução de overfitting**: modelos como XGBoost e LightGBM sao mais propensos a ajustar-se bem aos dados, no entanto o RandomForest e o LogisticRegression são mais robustos contra overfitting, levando a um equilibrio possível.
- **Meta modelo**: o nosso stacking utiliza LogisticRegression como modelo meta, aprendendo a combinar os outputs dos modelos base, o que reduz o risco de decisões mais viradas para um único modelo, o que melhora a **generalização**.

## 6.3 Método de treino

Relativamente ao treino, reservamos **80%** dos dados dividindo de forma estratificada devido ao desbalanceamento encontrado. Dentro desses 80%, aplicamos o *grid search* combinado com *cross validation* estratificado em 5 *folds* para otimizar os hiperparâmetros do modelo.

Importante referir que foi implementada uma abordagem personalizada de *cross-validation* com o objetivo de estimar métricas como a **variância** das previsões e um **bias** aproximado. Nesta abordagem, o conjunto de treino foi dividido em cinco *folds*, permitindo treinar e testar os modelos iterativamente em cada fold. Com isto, todas as previsões de cada *fold* foram recolhidas, proporcionando maior **flexibilidade** no cálculo das métricas.

Finalmente, os outros **20%** reservados exclusivamente para testes finais de modo a verificar a consistência de resultados e capacidade de generalização do modelo.

Através da seguinte figura é possível observar claramente a forma como dividimos o dataset principal e como foi aplicado o *cross validation* apenas no conjunto de treino.

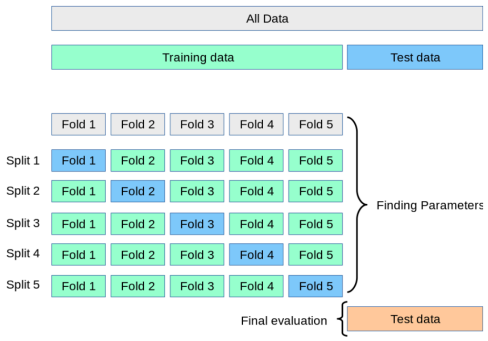



Figure 12: Cross Validation no conjunto de treino.

# 7 Análise dos resultados obtidos

A análise de resultados revelou-se um desafio maior do que o esperado, devido à inconsistência observada. Pequenas alterações na forma de treinar o modelo refletiram-se significativamente nos resultados.

Desta forma, procuramos encontrar o método mais justo de o fazer, analisando múltiplas métricas e iterando exaustivamente sobre os modelos.

## 7.1 Comparação de Resultados

Ao obter os resultados com validação cruzada, conseguimos perceber facilmente as vantagens de um bom **processamento** de dados e **otimização** de hiperparâmetros. Como mostra a figura seguinte, a azul observam-se os modelos iniciais, treinados com todas as features e sem otimização, enquanto à direita, a verde, estão os modelos finais, já otimizados, com uma melhoria geral de 5%.

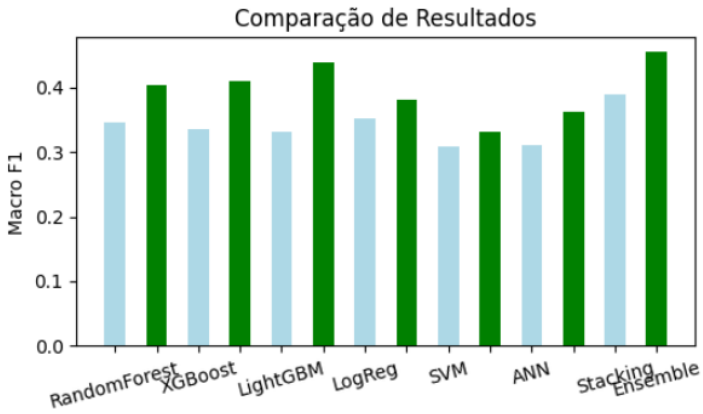



Figure 13: Comparação de resultados dos modelos selecionados

## 7.2 Avaliação de Métricas Adicionais

### 7.2.1 ROC e AUC

Para além dos resultados F1-macro obtidos com validação cruzada, analisámos também as curvas ROC e métricas AUC, como mostra a figura:

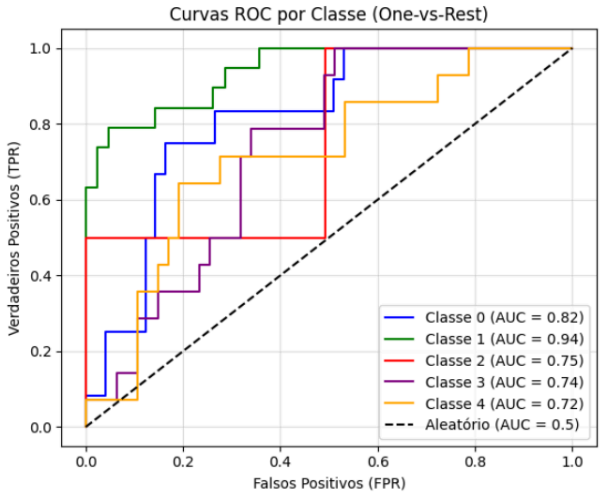



Figure 14: Curvas ROC e AUC

Estes dados foram obtidos com o modelo final escolhido: um **stacking ensemble** de cinco modelos diferentes — Random Forest, XGBoost, Logistic Regression, SVM e LightGBM.

Como é possível observar, o modelo poderia obter resultados mais otimistas para a Classe 2 (aquela com menos amostras). No entanto, isso implicaria um sacrifício nas outras classes, o que não considerámos vantajoso. Devido à existência de cinco classes, foi um desafio identificar a melhor abordagem para interpretar estes dados, pelo que nos concentrámos na melhoria geral da AUC.

### 7.2.2 Acertos por classe

Ainda dentro dos 20% dos dados para teste, procurámos obter a percentagem de acertos por classe, permitindo identificar quais necessitam de mais ou menos atenção. Os resultados estão ilustrados na seguinte figura:

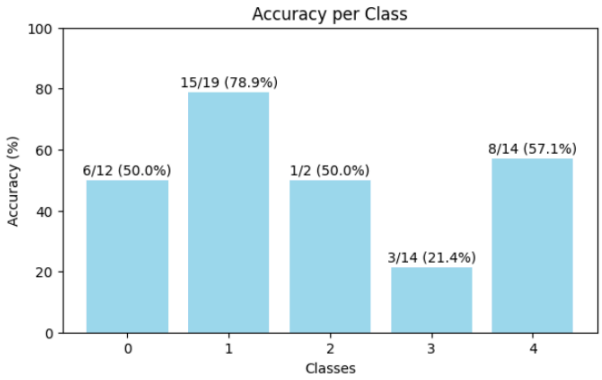



Figure 15: Accuracy por classe

### 7.2.3 Matriz de Confusão

A matriz de confusão fornece uma visão detalhada sobre as previsões, mostrando onde cada classe é bem ou mal prevista. Esta análise complementa o gráfico anterior.

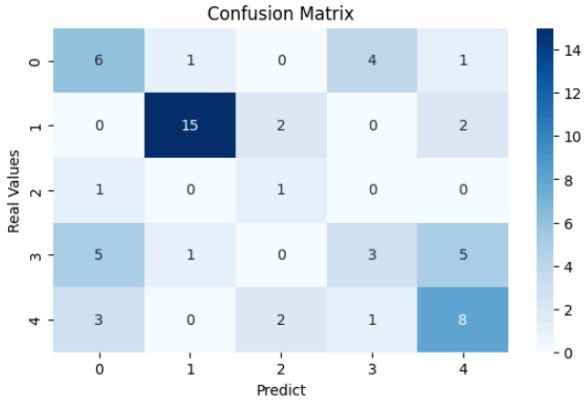



Figure 16: Matriz de Confusão

A partir destas métricas, identificámos que o modelo apresentava dificuldades em classes específicas, como a Classe 2 (CN-MCI), que possui menos amostras e, portanto, era mais difícil de prever, o que pode comprometer a aplicabilidade prática.

## 7.3 Testes finais

Como mencionado, **pequenas** variações no treino do modelo resultam em **grandes** diferenças nos resultados. Por isso, realizámos o máximo de testes possíveis, aplicando **diferentes** *seeds* na divisão do dataset e treinando os modelos com novos dados de treino, garantindo sempre que os dados de teste eram desconhecidos.

Para o modelo final, realizámos 10 testes com diferentes divisões, como mostra e exemplo:

```
7
F1 Macro Score: 0.4213
Bias: 0.786
Variance: 0.027
Ruído: 0.434
8
F1 Macro Score: 0.3709
Bias: 0.777
Variance: 0.03
Ruído: 0.439
9
F1 Macro Score: 0.408
Bias: 0.792
Variance: 0.03
Ruído: 0.51
```

Figure 17: Resultados para diferentes conjuntos de teste

Obtendo como média F1-macro dos 10 conjuntos um valor de **41%**, atingindo o objetivo proposto por nós de um F1-macro acima de 40%!

## 7.4 Hippocampo vs Lobo Occipital

Concluímos que, neste caso, a região do hipocampo está muito mais associada ao avanço da doença do que a região do lobo occipital. Isso reflete-se claramente nos resultados dos modelos, como mostra o exemplo:

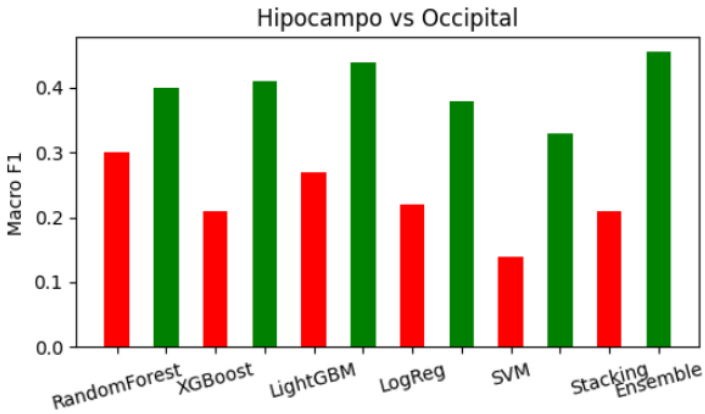



Figure 18: Resultados do Hipocampo vs Lobo Occipital

É notória a diferença de desempenho entre os modelos treinados com dados do hipocampo (verde) e do lobo occipital (vermelho), confirmando a maior relevância do hipocampo para este estudo.

# 8 Conclusão

Neste trabalho exploramos o uso de **Machine Learning** para prever a progressão de MCI para AD, com foco na importância das características radiômicas do hipocampo e do lobo occipital. Através de uma abordagem iterativa, desde a preparação de dados até à modelagem, conseguimos identificar que o **hipocampo** é significativamente **mais informativo** para este problema.

A utilização de técnicas como a **seleção de features** com SHAP Values e o **ensemble stacking** permitiram-nos atingir um desempenho bastante satisfatório, com um F1-macro **acima** de 40%. Apesar disso, algumas classes, como a Classe 2 (CN-MCI), apresentaram desafios devido ao **desbalanceamento** de dados, revelando a necessidade de mais amostras para melhorar a **generalização**.

Num trabalho futuro, seria esperado que melhorássemos o modelo para prever determinadas classes de acordo com a sua **urgência clínica**, **deixando de lado** a sua **performance** geral e **focando-se** no **perigo** representado para o paciente.

Em suma, este estudo contribuiu para a compreensão das relações entre alterações cerebrais e a progressão do Alzheimer, além de destacar o **potencial** de métodos avançados de **aprendizagem automática** no diagnóstico **precoce**.