Universidade do Minho

Departamento de Informática

Dados e Aprendizagem Automática

Design and optimization of Machine Learning models

Grupo 34

Gonçalo Araújo Brandão pg57874
Maya Gomes pg57891
Luís de Castro Rodrigues Caetano pg57887

26 de dezembro de 2025

Conteúdo

1 Business Understanding

O envelhecimento da população mundial tem gerado um aumento significativo nos casos de comprometimento cognitivo leve (MCI) , especialmente a Doença de Alzheimer (AD). A progressão de MCI para AD tem impactos devastadores na qualidade de vida dos pacientes e em seus cuidadores, além de gerar altos custos de tratamento.

A tarefa principal deste estudo é prever a progressão de MCI para AD, usando dados radiômicos extraídos de imagens de ressonância magnética (MRI). Esses dados oferecem informações detalhadas sobre texturas e distribuições de sinal em regiões específicas do cérebro, permitindo que modelos de aprendizado de máquina identifiquem padrões relevantes para a classificação.

2 Data Understanding

Como o Dataset fornecido pela equipa docente era muito vasto, o grupo decidiu analisá-lo com a utilização de CSV, obtendo os resultados que podemos visualizar na Tabela 1.

Assim, pela aplicação da biblioteca Python PyRadiomics, percebemos que o dataset estava dividido por grupos.

Grupos

Coluna 1	**Coluna 2**	**Coluna 3**
original	wavelet-LLH	wavelet-HHL
wavelet-LHL	wavelet-LHH	wavelet-HHH
wavelet-HLL	wavelet-HLH	wavelet-LLL
log-sigma-1-0	log-sigma-2-0	log-sigma-3-0
log-sigma-4-0	log-sigma-5-0	square_
square_	squareroot	logarithm
exponential	gradient	lbp-2D
lbp-3D-k	lbp-3D-m1	lbp-3D-m2

Tabela 1: Divisão de grupos no dataset.

Cada grupo está dividido em vários subgrupos, sendo que cada subgrupo representa uma coluna no nosso dataset.

Categorias de Features

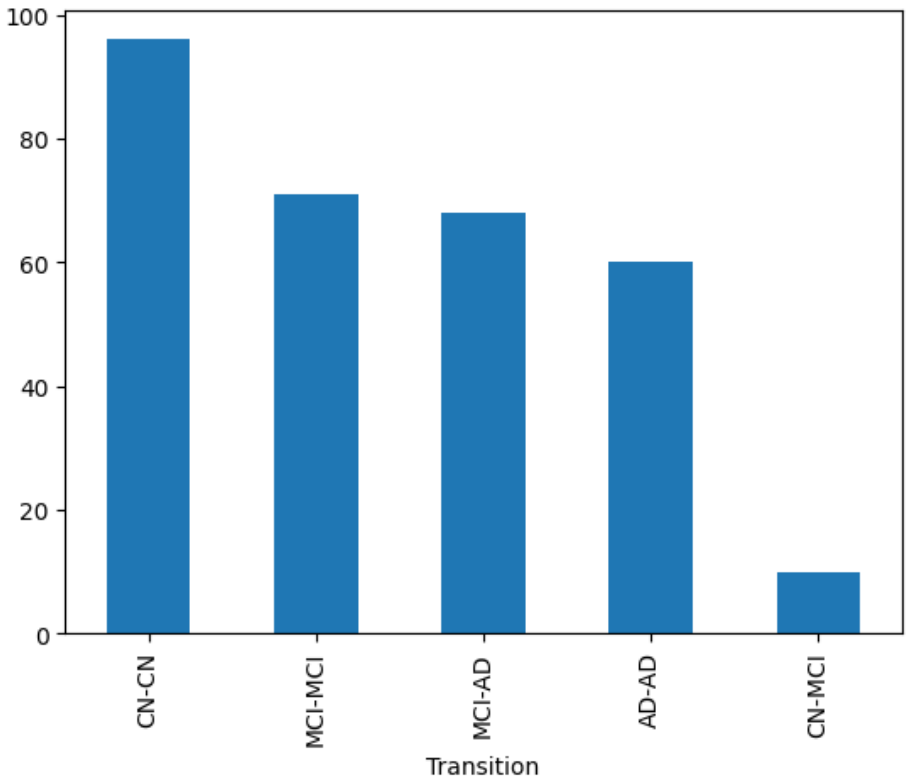
Categoria	**Descrição**
First Order Statistics	Extrai features estatísticas.
Shape-based (3D)	Captura características geométricas 3D.
Shape-based (2D)	Captura características geométricas 2D.
Gray Level Co-occurrence Matrix	Mede relações espaciais entre níveis de cinza.
Gray Level Run Length Matrix	Analisa sequências de níveis de cinza similares.
Gray Level Size Zone Matrix	Examina zonas de níveis de cinza similares.
Neighbouring Gray Tone Difference Matrix	Avalia diferenças entre vizinhos.
Gray Level Dependence Matrix	Captura dependências de níveis de cinza.

Tabela 2: Categorias de features radiômicas do dataset.

3 Data Exploration

O Data Exploration inicial dos Datasets relativos tanto ao Hippocampus como o Occipital revelaram as seguintes informações:

- **Número de Pacientes:** Os datasets contém um total de 305 linhas, cada uma representando um paciente único.
- **Número de Atributos:** Os datasets são compostos por 2181 atributos, representando as diferentes *features* extraídas das imagens médicas.
- **Null Values:** Não foram identificados valores nulos (`Null Values`) nos dataset.
- **Missing Values:** Não foram encontrados valores ausentes (*missing values*).
- **Valores Duplicados:** Não foram encontrados valores duplicados.
- **Análise do Target:** Analisando o target, percebemos a seguinte distribuição para ambos os datasets:
  - **CN-CN**: 96 amostras
  - **MCI-MCI**: 71 amostras
  - **MCI-AD**: 68 amostras
  - **AD-AD**: 60 amostras
  - **CN-MCI**: 10 amostras



4 Explore Data

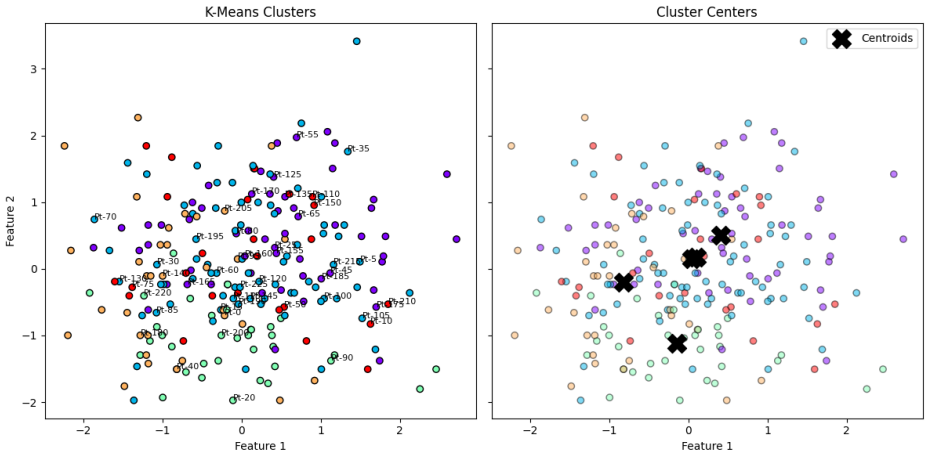
À semelhança do Data Exploration, o Explore Data, também apresenta resultados iguais para os dois datasets.

- **Remoção de Single-Value Columns:** Ao remover as Single-Value Columns, retiramos 159 colunas do nosso dataset.
- **Binerização da Feature "Age"**: Como os valores da "Age" eram do tipo int64, com valores como, 84.4 ou 74.3, decidimos realizar a Binerização dos mesmo. Aplicando bins nos seguintes intervalos, $[0, 65]$, $[65, 75]$, $[75, 85]$, $[85, 100]$, substituindo pelo valor medio do intervalo, [60, 70, 80, 90]
- **Remoção dos Dados Categóricos:** Analisando os dados Categóricos do Dataset, como 'ID', 'Image', 'Mask', 'diagnostics_Image-original_Hash', etc. Percebemos que estes não traziam nada de concreto ao nosso problema, sendo estes removidos .
- **Label Encoding:** Por fim, dicidimos realizar label enconding no nosso target.
  - **CN-CN**: 0
  - **AD-AD**: 1
  - **MCI-AD**: 2
  - **MCI-MCI**: 3
  - **CN-MCI**: 4
- **Análise Skewness:**
  - **Positive skewness**: 601
  - **Negative skewness**: 112
  - **Zero skewness**: 1448
- **Análise Kurtosis:**
  - **Positive kurtosis**: 969
  - **Negative kurtosis**: 118
  - **Zero kurtosis**: 1074

5 Modeling

Como escolha para o tipo de aprendizagem, para este tipo de problema, poderíamos ter utilizado Aprendizagem Supervisionada ou aprendizagem Não Supervisionada.

Para a Aprendizagem não supervisionada ser viável, seria necessário ao realizarmos, por exemplo o K-mean Clusters, iríamos encontrar alguns clusters das varias features, sendo possível encontrar um padrão. Ao realizar este teste obtivemos os resultados que são possíveis constatar na secção 5.1.



Como, analisando a imagem, percebemos que os cluster centers estão muito proximos, sendo que dois deles estão práticamente sobrepostos e as várias features encontra-se bastante misturadas, percebemos que a Aprendizagem não Supervisionada não seria uma opção.

Portanto, após decidirmos que iriamos utilizar Aprendizagem supervisionada, decidimos utilizar uma ferramenta da biblioteca *PyCaret* para testar quais seriam os melhores modelos a explorar.

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
rf	Random Forest Classifier	0.4564	0.7283	0.4564	0.4128	0.4154	0.2555	0.2668	0.0980
et	Extra Trees Classifier	0.4510	0.7316	0.4510	0.4069	0.4012	0.2441	0.2596	0.0930
knn	K Neighbors Classifier	0.4414	0.6573	0.4414	0.4240	0.4117	0.2398	0.2494	0.1750
xgboost	Extreme Gradient Boosting	0.4262	0.7023	0.4262	0.4369	0.4041	0.2170	0.2273	2.4260
gbc	Gradient Boosting Classifier	0.4221	0.0000	0.4221	0.4235	0.4023	0.2149	0.2225	4.0800
lightgbm	Light Gradient Boosting Machine	0.4167	0.6843	0.4167	0.4012	0.3922	0.2037	0.2126	2.4380
nb	Naive Bayes	0.4162	0.7143	0.4162	0.4135	0.3962	0.2145	0.2214	0.0740
lda	Linear Discriminant Analysis	0.3890	0.0000	0.3890	0.3871	0.3759	0.1786	0.1836	0.0840
lr	Logistic Regression	0.3743	0.0000	0.3743	0.3960	0.3685	0.1588	0.1632	0.8890
ada	Ada Boost Classifier	0.3438	0.0000	0.3438	0.3705	0.3333	0.1177	0.1222	0.2830
dt	Decision Tree Classifier	0.3345	0.5500	0.3345	0.3240	0.3193	0.1010	0.1040	0.0830
qda	Quadratic Discriminant Analysis	0.3260	0.0000	0.3260	0.2645	0.2654	0.0624	0.0682	0.0740
dummy	Dummy Classifier	0.3255	0.5000	0.3255	0.1065	0.1604	0.0000	0.0000	0.0670
svm	SVM - Linear Kernel	0.2964	0.0000	0.2964	0.1231	0.1629	0.0420	0.0537	0.0820
ridge	Ridge Classifier	0.2664	0.0000	0.2664	0.2971	0.2604	0.0205	0.0228	0.0660

Com o tratamento atual, o pycaret diz que os melhores modelos são :

- Random Forest Classifier
- Extra Trees Classifier
- Gradient Boosting Classifier
- K Neighbors Classifier
- Extreme Gradient Boosting

Assim decidimos utilizar estes 5 modelos e o Support Vector Classification nos nossos testes.

5.1 Testes Pré_Processemanto:

Para Realizarmos os testes para o pré_processemento realizamos o seguinte train_test_split():

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=34, stratify=y
)
```

Utilizamos o stratify para garantir que a mesma proporção de classes tanto nos dados de treino como nos dados de teste.
Com isto, criámos o cenário zero: com apenas as mudanças do pré_processamento, obtendo os seguintes resultados.

Modelo	**Accuracy**	**F1-Macro**	**Cross-Validation (F1-Macro)**
Extreme Gradient Boosting	**0.4156**	**0.3150**	0.2940
Gradient Boosting Classifier	0.3896	0.3046	**0.3142**
Support Vector Classification	0.3896	0.2548	0.2940
Random Forest Classifier	0.3766	0.2802	0.3318
Extra Trees Classifier	0.3766	0.2698	0.2979
K Neighbors Classifier	0.3636	0.2532	0.3359

5.2 F1-Macro

É importante destacar que o F1-Macro é calculado pela seguinte fórmula:

$$F1_{\text{macro}} = \frac{1}{C} \sum_{i=1}^{C} F1_i$$

com

$$F1_i = \frac{2 \cdot \text{Precision}_i \cdot \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i},$$

onde:

$$\text{Precision}_i = \frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i}, \quad \text{Recall}_i = \frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i}.$$

C = Número total de classes, TP_i = Verdadeiros Positivos da classe i,
FP_i = Falsos Positivos da classe i, FN_i = Falsos Negativos da classe i.

No caso da transição CN-MCI, existem apenas 10 amostras disponíveis. Devido ao número reduzido de exemplos, qualquer previsão correta para esta classe irá possuir um impacto significativo no cálculo do F1-Macro. Como o F1-Macro considera a média não ponderada entre as classes, uma única predição correta pode levar a um aumento expressivo no valor do F1-Macro.
Essa característica pode introduzir uma grande discrepância entre os resultados obtidos no conjunto de validação e no conjunto de teste, especialmente em cenários com distribuições desbalanceadas.

6 Feature Reduction

Como o dataset apresentado possui, após o pré-processamento, um shape de (305, 2014), apresentando 2014 Features, decidimos que o caminho mais apropriado seria realizar uma redução das mesmas. Há várias formas de fazer isto, como, por exemplo, por Principal Component Analysis (PCA), eliminação de Colunas Altamento Correlacionas, etc. O método escolhido nesta situação foi a eliminação de colunas altamente correlacionadas e colunas com um baixo desvio padrão.

6.1 Remoção de Colunas Altamente Correlacionadas

Para a remoção destes features decidimos testar vários Thresholds simultaneamente e ver em quais os nossos vários modelos obtinham os melhores resultados. Utilizamos a métrica do F1_Macro para decidir qual seria o melhor Threshold para cada algoritmo em análise.

Os Threshold utilizados foram de 0.80 a 1. A cada par de colunas cujo a sua correlação era superior ao Threshold, era mantida a coluna com maior correlação com o target.

Modelo	F1_Score (Original)	F1_Score (0.80)	F1_Score (0.81)	F1_Score (0.82)	F1_Score (0.83)	F1_Score (0.84)	F1_Score (0.85)	F1_Score (0.86)	F1_Score (0.87)
ExtraTrees	0.2698	0.3547	0.3241	0.3146	0.2502	0.3355	0.3128	0.2614	0.3698
RandomForest	0.2802	0.3513	**0.3800**	0.3789	0.3380	0.2951	0.2667	0.3354	0.3213
GradientBoost	0.3046	0.3643	0.3098	0.3588	**0.4097**	0.3355	0.3566	0.3542	0.3220
KNeighbors	0.2532	0.2649	0.2965	0.2939	0.2584	**0.3014**	**0.3014**	0.2600	0.2600
SVC	0.2548	0.3081	**0.3081**	0.3081	0.2678	0.2571	0.2571	0.2571	0.2571
XGBOOST	0.3150	0.3449	0.3451	**0.3705**	0.3344	0.3551	0.3271	0.3669	0.3431

Modelo	F1_Score F1_Score (Original)	F1_Score (0.88)	F1_Score (0.89)	F1_Score (0.90)	F1_Score (0.91)	F1_Score (0.92)	F1_Score (0.93)	F1_Score (0.94)	F1_Score (0.95)	F1_Score (0.96)
ExtraTrees	0.2698	0.3046	0.2794	0.3526	0.2866	0.2846	**0.3707**	0.2544	0.3356	0.3118
RandomForest	0.2802	0.3686	0.3235	0.3422	0.3169	0.2633	0.3439	0.2816	0.2976	0.2685
GradientBoost	0.3046	0.3031	0.3637	0.3152	0.3138	0.2973	0.3127	0.2923	0.2900	0.2971
KNeighbors	0.2532	0.2600	0.2600	0.2444	0.2444	0.2444	0.2308	0.2308	0.2311	0.2311
SVC	0.2548	0.2571	0.2571	0.2571	0.2571	0.2571	0.2571	0.2571	0.2571	0.2571
XGBOOST	0.3150	0.3697	0.3447	0.3448	0.3194	0.3393	0.3181	0.2812	0.2788	0.3068

Resumo Dos Resultados Obtidos

Modelo	F1_Score (Original)	Melhor Threshold	Melhor F1_Score	Melhoria
ExtraTrees	0.2698	0.93	0.3707	+ 0.1009
RandomForest	0.2802	0.81	0.3800	+ 0.0998
GradientBoost	0.3046	0.83	0.4097	+ 0.1041
KNeighbors	0.2532	0.84	0.3014	+ 0.0482
SVC	0.2548	0.81	0.3081	+ 0.0533
XGBOOST	0.3150	0.82	0.3705	+ 0.0555

Como os algoritmos KNeighbors e SVC não apresentão resultados comparáveis aos outros modelos, decidimos não utiliza-los mais nos nossos testes seguintes

Percebemos assim que vários Thresholds iriam beneficiar os modelos de forma diferente, por tanto, decidimos criar um dataset para cada modelo, consoante o resultado do seu melhor Threshold. Também é importante realçar que estes resultados são influenciados pelo nosso train_test_split.

7 Remoção de Feature Com Baixo Desvio Padrão

Também achámos pertinente remover as features que sobravam em cada dataset de cada modelo e aquelas que possuíam desvio padrão bastante baixo, pois estas são muito similares a single value columns.

Utilizámos a mesma metodologia que utilizamos na remoção de colunas altamente correlacionadas e testamos thredholds entre 0.001 a 0.01, sendo que os resultados mais relevantes foram obtidos entre 0.001 a 0.005

Modelo	F1_Macro Original	F1_Macro (0.001)	F1_Macro (0.002)	F1_Macro (0.003)	F1_Macro (0.004)	F1_Macro (0.005)
Random Forest	0.3800	0.3276	0.3037	0.3980	0.2766	**0.4198**
Extra Trees	0.3707	0.3313	0.3168	0.2567	0.3076	0.2759
Gradient Boost	0.4097	0.4057	0.3644	0.3421	0.3642	0.3395
XGBoost	0.3705	0.3478	0.3452	0.3196	0.3266	0.3214

Nestes intervalos temos que o único modelo que beneficiou desta remoção foi o Random_Forest, sendo esta feature reduction apenas utilizada neste algoritmo.

8 Outliers

Com a redução do número de features e o respetivo tamanho do dataset, decidimos analisar os outliers e tratá-los de forma a melhorar os nossos modelos, mesmo que alguns já fossem robustos a outliers.

Como termo de comparação decidimos testar para o dataset antes da feature reduction. Sendo que o método utilizado começa por identificar os outliers no dataset e usa o IQR (Interquartile Range) e analisando a sua distribuição. Para cada coluna, calcula-se os limites de outliers e identifica-se as linhas fora desses limites.

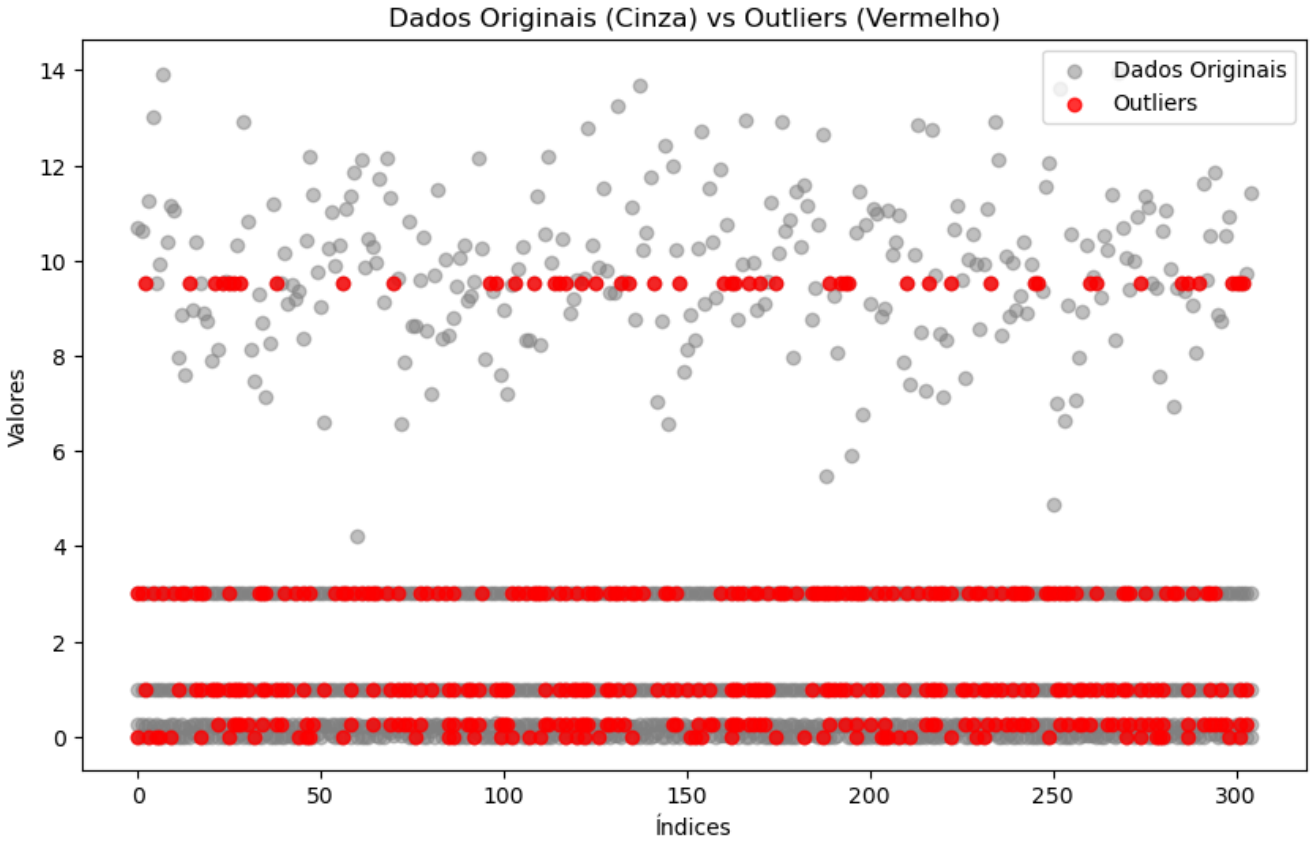
Com base nos outliers detetados, calculamos o número e a percentagem de outliers em relação ao total de categoria de cada classe, além de determinar a percentagem de cada classe no conjunto total de outliers. O resultado é um resumo da distribuição de outliers por classe.

```
(305, 2014)
   Class  Outliers Count  Total Outliers  Percentage of Total Outliers
0      0              96             305                         100.0
1      3              71             305                         100.0
2      2              68             305                         100.0
3      1              60             305                         100.0
4      4              10             305                         100.0
```

Testando para um dataset reduzido

```
(305, 385)
   Class  Outliers Count  Total Outliers  Percentage of Total Outliers
0      0              95             303                     98.958333
1      3              71             303                    100.000000
2      2              67             303                     98.529412
3      1              60             303                    100.000000
4      4              10             303                    100.000000
```

Com estes testes percebemos que, pelo o menos uma vez, cada linha do nosso dataset era identificada com outliers nas suas features e isto era verdade tanto para o dataset após o processamento e após a redução de features. Com estes resultados decidimos sobrepor todos os pontos do nosso dataset (a cinzento) e os outliers (a vermelho) para termos uma melhor perceção do problema.



Assim percebemos que estes eram parte fundamental do nosso dataset e que estes eram transversais a classes e a features, portanto decidimos não os tratar na nossa resolução, pois estes continham informação sobre o problema.

9 Modelos

9.0.1 Processamento:

Após lidarmos o dataset de muitas maneiras, decidimos forcar-nos nos modelos. Para tal decidir-mos analisar 3 parametros em cada modelo antes de realizar-mos o Hypertunning. Decidimos analisar o impacto da normalização, utilização do SMOTE, criando amostras sintéticas no dataset de treino e remover features com baixa feature importance. Utilizamos estas três metodologias nos 4 modelos que obtivemos melhores resultados até ao momento, Random Forest, Extra Trees, Gradient Boosting e XGBoost.

9.0.2 HyperTunning

Decidimos realizar o hypertunning com recurso a biblioteca optuna, utilizando o TPESmapler (Tree-structured Parzen Estimator) um metodo que se foca nos parâmetros que obtiveram melhores resultados e vai tentando maximizar os nosso resultados ao longo de várias tentativas.
Estas duas metodologias foram aplicadas aos 4 modelos de forma similar.

9.1 Random Forest

O Random Forest é um algoritmo de aprendizagem baseado em árvores de decisão que combina múltiplas árvores para criar um modelo mais robusto e preciso. Estes algoritmo é resistente a outliers, pois cada árvore toma decisões baseadas em divisões locais nos dados. Também não exige normalização ou padronização dos dados, contudo decidimos testar na mesma.

9.1.1 Resultados:

Todos os resultados devem ser consultados nos vários notebooks submitidos na entrega do trabalho, sendo que a métrica analisa sempre foi o f1_score_macro.

Modelo	F1-Score Inicial	Dados Normalizados	com SMOTE	Feature Importance
Random Forest	0.36209	0.36469	0.36209	0.31297

Tabela 3: Comparação de F1-Score para diferentes técnicas.

Analise de Resultados Processamento :

- A normalização trouxe um benefício muito ligeiro de 0.002 ao nosso modelo.
- A introdução de dados sintéticos não trouxe qualquer melhoria ao nosso modelo. Uma das possíveis causas pode ser a complexidade da criação de dados sintéticos a partir de dados biológicos, que frequentemente introduz ruído ao invés de valor.
- O uso de feature importance, testado para vários thresholds, teve impacto negativo. Isso indica que as colunas com menos importância ainda trazem informações valiosas ao modelo e não devem ser descartadas.

Analise de Resultados Hypertunning :

- **Melhores Hiperparâmetros:** { `n_estimators`: 143, `max_features`: `'sqrt'`, `max_depth`: 24, `min_samples_split`: 32, `min_samples_leaf`: 13, `bootstrap`: `False`, `class_weight`: `'balanced'` }
- **Cross Validation Score:** [0.47478261, 0.29848841, 0.4116514, 0.43780664, 0.35315827]
- **Média do Cross Validation Score:** 0.39517746656845654
- **Desvio Padrão do Cross Validation Score (STD):** 0.06251400064162042

9.2 XGBoost:

O XGBoost é um algoritmo de aprendizado supervisionado baseado em árvores de decisão, que utiliza um processo de boosting para melhorar o modelo, combinando várias árvores fracas em uma forte. Ele é bastante resistente aos outliers, pois as árvores de decisão tendem a ser menos sensíveis a valores extremos em comparação com outros algoritmos. Além disso, o XGBoost não exige normalização dos dados, pois as árvores de decisão funcionam bem independentemente da escala das variáveis, tornando o pré-processamento mais simples.

9.2.1 Resultados:

Modelo	F1-Score Inicial	Dados Normalizados	com SMOTE	Feature Importance
XGBoost	0.32926	0.33064	0.32926	0.30826

Tabela 4: Comparação de F1-Score para diferentes técnicas.

Analise de Resultados Processamento :

- A normalização trouxe um benefício muito ligeiro ao nosso modelo.
- A introdução de dados sintéticos não trouxe qualquer melhoria ao nosso modelo.
- O uso de feature importance, testado para vários thresholds, teve impacto negativo. Isso indica que as colunas com menos importância ainda trazem informações valiosas ao modelo e não devem ser descartadas.

Analise de Resultados Hypertunning :

- **Melhores Hiperparâmetros:** { `n_estimators`: 846, `max_depth`: 5, `learning_rate`: 0.1660773939789664, `subsample`: 0.9293091645826319, `colsample_bytree`: 0.5898888173202074, `gamma`: 0.0035106908632086685, `reg_alpha`: 3.3065806009993237e-06, `reg_lambda`: 0.6447155795041164, `min_child_weight`: 7 }
- **Cross Validation Score:** [0.34841315, 0.39514309, 0.4412588, 0.32624358, 0.3037629]
- **Média do Cross Validation Score:** 0.36296430450078626
- **Desvio Padrão do Cross Validation Score (STD):** 0.049457974494161415

9.3 Extra Trees

O Extra Trees é um algoritmo supervisionado baseado em árvores de decisão, que usa aleatoriedade na construção das árvores para melhorar a robustez. É resistente a outliers e não exige normalização, pois não depende da escala das variáveis.

9.3.1 Resultados:

Modelo	F1-Score Inicial	Dados Normalizados	com SMOTE	Feature Importance
Extra Trees	0.31493	0.32773	0.32773	0. 0.30819

Tabela 5: Comparação de F1-Score para diferentes técnicas.

Analise de Resultados Processamento :

- A normalização trouxe um benefício muito ligeiro ao nosso modelo.
- A introdução de dados sintéticos não trouxe qualquer melhoria ao nosso modelo.
- O uso de feature importance, testado para vários thresholds, teve impacto negativo no nosso modelo, logo não a vamos utilizar. Isso indica que as colunas com menos importância ainda trazem informações valiosas ao modelo e não devem ser descartadas.

Analise de Resultados Hypertunning :

- **Melhores Hiperparâmetros:** { `n_estimators`: 111, `max_depth`: 54, `min_sample_split`: 14, `min_samples_leaf`: 4, `bootsrap`: false, `class_weight`: balanced, `criterion`: gini, `max_features`: sqrt }
- **Cross Validation Score:** [0.39933952 0.31090838 0.43019421 0.4005176 0.29704433]
- **Média do Cross Validation Score:** 0.36760080952882745
- **Desvio Padrão do Cross Validation Score (STD):** 0.05329360632032132

9.4 Gradient Boosting:

O Gradient Boosting é um algoritmo supervisionado que combina várias árvores de decisão ajustadas iterativamente para minimizar o erro. É resistente a outliers em menor grau que outros métodos baseados em árvores, pois utiliza gradientes que podem ser influenciados por valores extremos. Não exige normalização, já que as árvores não dependem da escala das variáveis.

9.4.1 Resultados:

Modelo	F1-Score Inicial	Dados Normalizados	com SMOTE	Feature Importance
Gradient Boosting	0.31446	0.32247	0.32247	0.35

Tabela 6: Comparação de F1-Score para diferentes técnicas.

Analise de Resultados Processamento :

- A normalização trouxe um benefício ligeiro ao nosso modelo .
- A introdução de dados sintéticos não trouxe qualquer melhoria ao nosso modelo.
- O uso de feature importance, testado para vários thresholds, teve impacto de forma geral negativo, contudo em certos thresholds positivo. Isso indica que as colunas com menos importância ainda trazem informações valiosas ao modelo e não devem ser descartadas.

Analise de Resultados Hypertunning :

- **Melhores Hiperparâmetros:** { `n_estimators`: 109, `max_depth`: 3, `learning_rate`: 0.04471222414633008, `subsample`: 1.0, `min_samples_split`: 18, `min_samples_leaf`: 1, `max_features`: sqrt }
- **Cross Validation Score:** [0.37878788 0.42034146 0.246 0.44918551 0.37361576]
- **Média do Cross Validation Score:** 0.3735861230552431
- **Desvio Padrão do Cross Validation Score (STD):** 0.06957960480563609

9.5 Resultados Kaggle:

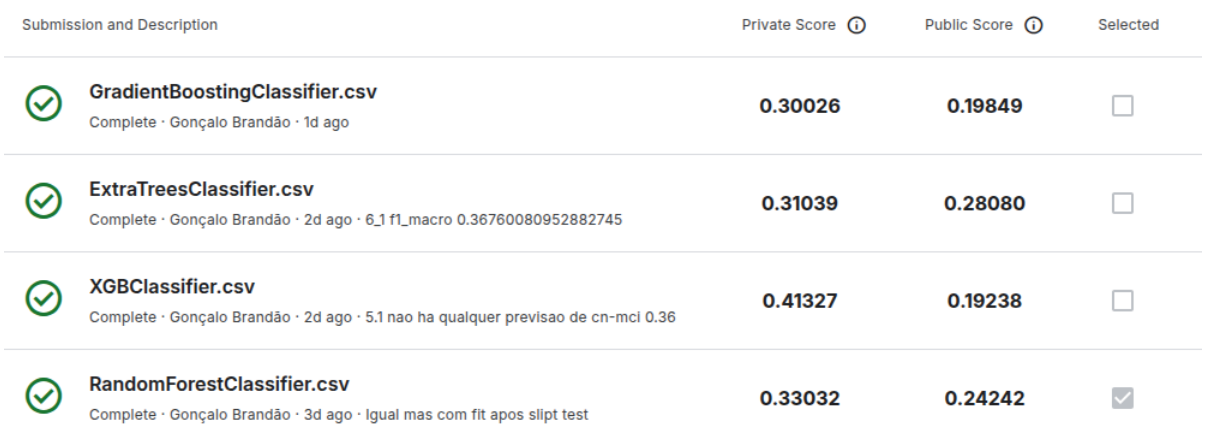


Figura 1: Resultados registados no Kaggle

Modelo	**Média**	**Desvio Padrão**	**Intervalo Possível**
Random Forest	0.3952	0.0625	[0.3327, 0.4577]
XGBoost	0.3630	0.0495	[0.3135, 0.4124]
ExtraTrees	0.3676	0.0533	[0.3143, 0.4209]
GradientBoosting	0.3736	0.0696	[0.3040, 0.4432]

Tabela 7: Resultados do Cross Validation Score para diferentes modelos.

Selecionamos o Random Forest, uma vez que localmente foi o modelo que nos deu melhores resultados, com um f1_score_macro de 0.39.

Observando os resultados do Kaggle estão dentro do esperado, pois todos os resultados se encontram dentro do desvio padrão da cross_validation_score.

O modelo por nós selecionado, o Random Forest, obteve uma classificação de 0.33032, garantindo nos o 36º lugar na competição. Assim, o nosso grupo desceu 14 lugares da pública para a privada.

Analisando os resultados, percebemos que tanto o Random Forest, como o ExtraTrees e o GradientBoosting, obtiveram classificações próximas do mínimo possível no intervalo esperado de resultados. Por outro lado, o XGBoost, aproximou-se do máximo previsto. Caso tivéssemos escolhido este modelo, teríamos ficado no top 5 dos resultados da competição.

10 Comparação de Resultados Com o Dataset de Controlo

Modelo	Média (Hippo)	Desvio Padrão (Hippo)	Média (Occipital)	Desvio Padrão (Occipital)	Diferença (Hippo - Occipital)
Random Forest	**0.3952**	**0.0625**	**0.2241**	**0.0392**	**0.1711**
XGBoost	**0.3630**	**0.0495**	**0.2345**	**0.03346**	**0.1285**
ExtraTrees	**0.3676**	**0.0533**	**0.2204**	**0.036**	**0.1472**
GradientBoost	**0.3736**	**0.0696**	**0.2375**	**0.038**	**0.1361**

Comparando os resultados obtidos entre o dataset de controlo e o nosso dataset de testes percebemos que o Hippocampus, sim tem mais relação com a Alzheimer's disease do que o Occipital. Contudo, no melhor caso obteve-se uma diferença de 17% e no pior caso de 12.8%.

Estes resultados demonstram que ambos os datasets não são bons para prever a Alzheimer's disease em pacientes, já que as previsões estão erradas em maior parte dos casos de estudo.

11 Outra Abordagem

Como durante o processo de desenvolvimento sentiamos que não conseguiamos progredir no projeto, quase como tivessemos atingido um limite superior, decidimos experimentar outra abordagem.

Como o Target é uma transição de um paciente de um estado Alzheimer's disease (AD) ou Mild Cognitive Impairment (MCI) ou Cognitive Normal (CN) para um dentes estados, sendo que só há 5 transições validas, CN-CN, CN-MCI, CN-AD, MCI-MCI, MCI-AD, e AD-AD.

O grupo decidiu dividir o target Transition em duas colunas, origem e destino e criar um modelo para prever a origem da transição e outro para prever o destino.

	Origem	Destino	Transition
0	CN	CN	CN-CN
1	CN	CN	CN-CN
2	AD	AD	AD-AD
3	CN	MCI	CN-MCI
4	CN	CN	CN-CN
..	...	...	...
300	CN	CN	CN-CN
301	CN	CN	CN-CN
302	MCI	AD	MCI-AD
303	MCI	MCI	MCI-MCI
304	CN	CN	CN-CN

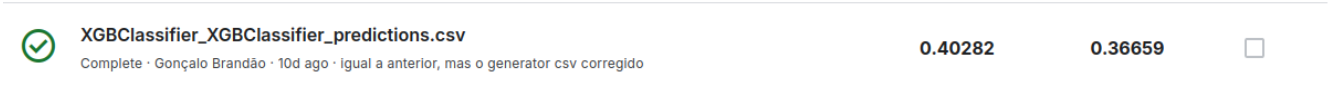
Depois utilizamos uma função auxiliar para unir as duas previsões, sendo que para as previsões impossíveis, eram reformuladas consoante os f1_score da origem e do destino. Isto é, por exemplo, se o modelo que prevê o destino, previa AD, a origem também teria de ser AD, pois é o único caso possível e a previsão do destino AD é a previsão com maior f1_score no modelo.

```
F1 Macro (Cross-Validation): 0.5187944570049834
Relatório de Classificação (Teste):
              precision    recall  f1-score   support

          AD       0.50      0.33      0.40        24
          CN       0.66      0.79      0.72        42
         MCI       0.57      0.57      0.57        56

    accuracy                           0.60       122
   macro avg       0.58      0.56      0.56       122
weighted avg       0.59      0.60      0.59       122

F1 Macro (Cross-Validation): 0.4843218957796963
Relatório de Classificação (Teste):
              precision    recall  f1-score   support

          AD       0.74      0.82      0.78        51
          CN       0.56      0.76      0.64        38
         MCI       0.38      0.15      0.22        33

    accuracy                           0.62       122
   macro avg       0.56      0.58      0.55       122
weighted avg       0.59      0.62      0.58       122
```

Algo que nos deu segurança nesta abordagem foi que para o dataset após o pré_processamento obtinhamos f1_macro de 0.50 tanto para origem como para o destino e um f1_score das previsões combinadas entre os 0.35 e 0.42. Também, foi possível começar a prever corretamente alguns casos da classe minoritária CN-MCI.

Contudo o grupo desistiu desta abordagem, pois o dataset foi feito para prever transições e estamos a prever a origem e o destino a partir do mesmo. Considerámos assim que estávamos a deturpar os dados e a concluir resultados não pretendidos para o dataset.

Mesmo assim, fizemos duas submissões no kaggle sendo que uma delas teve um score de 0.40 na privada, estando muito bem colocada na leaderboard.



12 Conclusão

Com a conclusão deste projeto, é pertinente oferecer uma reflexão crítica sobre o trabalho realizado, que nos proporcionou a oportunidade de consolidar os conhecimentos adquiridos na disciplina de DAA. Consolidamos conceitos relacionados com o tratamento de dados e à construção de modelos de aprendizado de máquina.

Neste trabalho, destacamos a utilização de diversos modelos de aprendizado de máquina e técnicas para o tratamento de dados, abrangendo desde o feature reduction até a normalização e criação de dados sintéticos, finalizando com a otimização de hiperparâmetros.

Por fim, acreditamos que este documento apresenta uma explicação clara e alinhada com as etapas da metodologia CRISP-DM. O conjunto de modelos desenvolvidos é abrangente e a análise realizada foi eficaz, fornecendo respostas corretas e consistentes para os problemas identificados no dataset. Em resumo, consideramos que o balanço do trabalho é positivo, superamos as dificuldades encontradas e cumprimos com os requisitos estabelecidos.