Universidade do Minho
Escola de Engenharia

Mestrado em Engenharia Informática

2024/2025

Dados e Aprendizagem Automática

Trabalho Prático

Grupo 21

João Rodrigues - pg57880
Tiago Rodrigues - pg56013
Marcelo Sousa - pg57584
José Pacheco - pg55972

Índice

1 Introdução

Este trabalho prático centra-se no desenvolvimento de modelos de Aprendizagem Automática para analisar a evolução do Declínio Cognitivo Ligeiro (MCI) e prever a sua progressão para a Doença de Alzheimer (AD), uma condição neurodegenerativa que afeta milhões de pessoas em todo o mundo.

O estudo utiliza dados de ressonância magnética (MRI) e técnicas de radiomics, que permitem a extração de características quantitativas a partir de imagens médicas. A análise incide sobre regiões específicas do cérebro, nomeadamente o hipocampo, fundamental na investigação sobre Alzheimer, e o lobo occipital, usado como controle. Através da aplicação de métodos de aprendizagem automática, pretende-se identificar padrões relevantes que auxiliem na previsão da progressão do MCI, contribuindo para uma detecção precoce e possíveis intervenções.

Este documento apresenta os objetivos, a metodologia e os resultados obtidos ao longo do desenvolvimento do trabalho, fornecendo ainda uma análise crítica sobre a aplicação dos modelos e a sua relevância.

2 Metodologia

Para alcançar os objetivos definidos, foi seguida uma abordagem estruturada dividida em várias etapas fundamentais:

- **Definição do problema:** Inicialmente, foi realizada uma compreensão aprofundada do problema, identificando os principais desafios e requisitos do projeto.
- **Análise e Exploração dos Dados:** Nesta fase, foi conduzida uma análise exploratória detalhada para compreender as principais características do conjunto de dados. Esta etapa permitiu definir estratégias de limpeza e seleção de atributos, visando melhorar a qualidade dos dados utilizados na modelação.
- **Preparação dos Dados:** A preparação dos dados envolveu diversas técnicas de processamento e transformação, garantindo que os dados estivessem adequadamente estruturados para a aplicação dos modelos de aprendizagem.
- **Modelação:** Foram testados diferentes algoritmos de aprendizagem automática para a modelação. A seleção dos modelos baseou-se em experiências prévias e na otimização de hiperparâmetros, realizada por meio de técnicas como grid search.
- **Avaliação do modelo candidato:** O desempenho dos modelos foi avaliado utilizando métricas adequadas, com ênfase na F1-score macro para lidar com o desbalanceamento das classes. Além disso, foi utilizado um conjunto de dados de controlo para verificar a presença de anomalias ou tendências inesperadas.
- **Submissão no Kaggle:** Os resultados do modelo foram submetidos ao Kaggle, permitindo a comparação com os desempenhos obtidos durante a fase de treino. Esta comparação possibilitou validar a capacidade de generalização do modelo e identificar potenciais problemas de overfitting ou underfitting.

3 Definição do problema

O objetivo deste projeto é desenvolver modelos de aprendizagem automática para prever a progressão do Declínio Cognitivo Ligeiro (MCI) para a Doença de Alzheimer (AD), utilizando dados de ressonância magnética (MRI).

O foco está na análise de características extraídas de regiões cerebrais específicas, como o hipocampo, uma região essencial para o diagnóstico da Alzheimer. Para isso, dispomos de um conjunto de dados de treino, que será utilizado para identificar o modelo com o melhor desempenho.

Para além dos conjuntos de dados de treino e teste, foi-nos fornecido um conjunto de dados de controlo referente a outra região do cérebro humano. Este conjunto serve para verificar se existem anomalias no funcionamento do modelo, tais como vazamento de dados ou excesso de ajuste ("overfitting"), que poderiam comprometer a capacidade de generalização do modelo a novos dados.

Dado que a variável que pretendemos prever, denominada "Transition", é uma variável categórica, trata-se de um problema de classificação. Assim, serão exploradas técnicas e algoritmos apropriados para esse tipo de tarefa.

4 Análise dos dados

Para uma melhor compreensão do problema e do conjunto de dados, foi realizada uma análise exploratória detalhada. Este processo permitiu identificar características relevantes dos dados, bem como possíveis desafios associados ao seu uso em modelos de Aprendizagem Automática.

4.1 Dimensionalidade e Estrutura

O conjunto de dados de treino analisado continha 305 linhas e 2181 colunas, o que reflete uma grande dimensão. Este elevado número de características exige uma análise para identificar quais delas são relevantes para o problema em questão. Durante esta análise inicial, foi constatado que não havia valores em falta (missing values), o que eliminou a necessidade de aplicar técnicas de imputação ou substituição de valores ausentes.

4.2 Colunas Constantes e Irrelevantes

Identificámos a existência de 159 colunas constantes, ou seja, variáveis cujos valores eram idênticos em todas as observações. Estas colunas não contêm informação útil para o modelo, pelo que foram consideradas irrelevantes e marcadas para exclusão. Além disso, algumas variáveis categóricas relacionadas com referências às imagens no dataset foram avaliadas como não pertinentes para os objetivos do projeto e também foram descartadas.

4.3 Variável Dependente

A variável dependente, Transition, que descreve as transições entre estados cognitivos, revelou-se uma variável categórica desbalanceada. Este desequilíbrio pode dificultar o desempenho dos modelos de classificação, uma vez que os algoritmos podem tender a favorecer a classe maioritária.

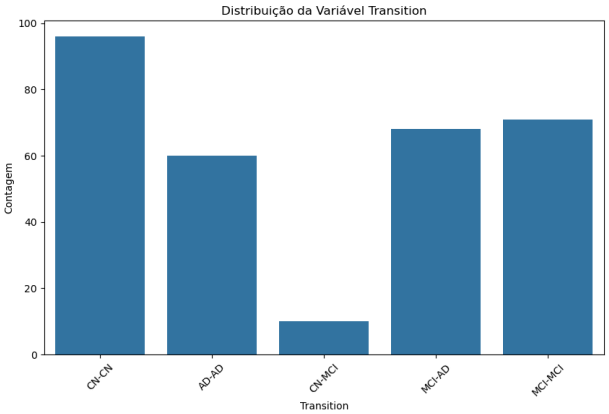

Figura 2: Distribuição da variável Transition

4.4 Conclusão da análise

A análise preliminar dos dados revelou pontos-chave para a preparação dos mesmos, nomeadamente:

– A exclusão de colunas constantes e variáveis irrelevantes.

– A necessidade de tratar o desequilíbrio da variável dependente.

– A alta dimensão do dataset, que exige métodos eficazes de seleção ou redução de features.

5 Tratamento de Dados

O tratamento dos dados é uma etapa fundamental no processo de aprendizagem automática. Nesta fase, testámos diferentes abordagens de preparação de daos com o objetivo de identificar a melhor estratégia para otimizar o desempenho dos modelos na fase de modelação.

5.1 Preparação Inicial

Na fase inicial, realizou-se uma análise preliminar dos dados para identificar possíveis problemas e definir as estratégias de tratamento necessárias. Esta fase serviu como base para as etapas subsequentes do processo de limpeza e transformação dos dados.

5.2 Remoção de Colunas Constantes

Umas das etapas da preparação consistiu na identificação e remoção de colunas constantes do conjunto de dados. Estas colunas, por não apresentarem variação nos seus valores, não contribuem para o poder preditivo dos modelos e podem ser eliminadas sem perda de informação relevante.

5.3 Remoção de colunas identificadoras

Foi realizada a remoção de colunas identificadoras, tais como IDs, URLs de imagens e outras variáveis categóricas que não continham informações relevantes para o objetivo do projeto. Estas colunas, apesar de úteis para referência, não possuem impacto direto na tarefa de previsão.

5.4 Análise de Correlação

Para reduzir a redundância nos dados, realizámos uma análise de correlação entre as variáveis, identificando e removendo colunas altamente correlacionadas (correlação > 0.9). Em algumas preparações específicas, adotou-se um limiar mais restritivo (correlação > 0.8).

5.5 Tratamento de Variáveis Categóricas

As colunas categóricas do conjunto de dados foram tratadas de duas formas distintas:

- Remoção completa: Todas as colunas categóricas foram removidas, considerando a possibilidade de que não contribuiriam de forma significativa para a previsão.
- Transformação de categóricas em numéricas: Algumas colunas categóricas foram convertidas para formato numérico. No dataset original, estas colunas encontravam-se representadas como tuplos, exigindo uma prévia transformação e extração adequada dos valores.

5.6 Normalização dos Dados

Para garantir que todas as variáveis contribuam de forma equilibrada para os modelos, aplicámos duas técnicas distintas de normalização:

5.6.1 Normalização Min-Max

Esta técnica foi aplicada para escalar os dados para um intervalo específico, entre 0 e 1, mantendo a distribuição relativa dos valores originais.

5.6.2 Normalização Z-Score

Também conhecida como padronização, esta técnica transforma os dados para que tenham média zero e desvio padrão igual a 1, sendo particularmente útil quando os dados seguem uma distribuição aproximadamente normal.

5.7 Diferentes Abordagens de Preparação

Foram consideradas várias combinações das técnicas de pré-processamento mencionadas, resultando em diferentes configurações de preparação dos dados, conforme ilustrado na tabela abaixo:

Prep.	Remover colunas constantes	Remover colunas com alta Correlação	Remover colunas Categóricas	Transformação de categóricas em numéricas	Min-Max	Z-Score
1	✓	>0.9	✓			
2	✓	>0.9	✓		✓	
2b	✓	>0.8	✓		✓	
3			✓		✓	
4			✓			✓
5				✓	✓	

Tabela 1: Diferentes configurações de preparação de dados

5.8 Técnicas de Balanceamento de Classes

Como visto na análise de dados existe um certo desbalanceamento na variável dependente, dessa forma para cada preparação de dados testamos também técnicas de balanceamento, como:

- **SMOTE (Synthetic Minority Over-sampling Technique):** para gerar novas instâncias sintéticas da classe minoritária.
- **Random OverSampling:** Aumenta a representação da classe minoritária por meio da repetição aleatória de instâncias existentes.

5.9 Redução de Dimensionalidade

Adicionalmente, foram utilizadas técnicas de redução de dimensionalidade, como a Análise de Componentes Principais (PCA), para diminuir o número de features mantendo a máxima variância explicada.

6 Modelação

Com as diferentes preparações realizadas, a próxima etapa consiste em aplicá-las a diversos modelos de aprendizagem automática para avaliar o seu desempenho.

O foco principal foi a utilização de algoritmos de aprendizagem supervisionada. Para cada algoritmo, implementámos uma abordagem de grid search combinada com validação cruzada (cross-validation) com k = 10, a fim de encontrar a melhor combinação de hiperparâmetros e garantir a robustez dos modelos.

Após a identificação do melhor modelo, realizámos uma validação adicional utilizando a abordagem de hold-out validation, com uma divisão de 80% dos dados para treino e 20% para teste. Esta técnica permitiu obter a matriz de classificação e comparar os resultados obtidos com a validação cruzada, assegurando a generalização do modelo.

Após a seleção do melhor modelo, procedemos à avaliação utilizando o dataset de controlo, que, como mencionado anteriormente, contém dados de uma região diferente do cérebro. O objetivo desta etapa é verificar a robustez do modelo e a sua capacidade de generalização. Espera-se que o valor da métrica F1 neste dataset seja próximo ou inferior ao valor de "pure chance", que corresponde à probabilidade de acertar aleatoriamente numa classe. Como a variável dependente possui cinco classes, a probabilidade de classificação aleatória é de 1/5 = 0.20 (20%). Caso o modelo obtenha um valor significativamente superior a este limiar no dataset de controlo, isso pode indicar a existência de vazamento de dados (data leakage) ou overfitting, sugerindo que o modelo pode estar a capturar padrões artificiais presentes apenas nos dados de treino. Assim, esta verificação permite-nos controlar e validar a fiabilidade do modelo antes da sua aplicação prática.

Por fim, os resultados foram submetidos na plataforma Kaggle, onde analisámos o desempenho do modelo nas partições pública do conjunto de testes fornecido. A comparação entre os scores públicos e do treino permitiu identificar se o modelo apresentava sinais de overfitting ou underfitting.

Em seguida, apresentamos os algoritmos explorados e os respetivos resultados dos modelos.

6.1 Decision Tree

A Decision Tree é um algoritmo de classificação baseado numa estrutura de árvore de decisão, onde os dados são divididos recursivamente em função de regras baseadas nas características do conjunto de dados. Os parâmetros que utilizamos foram os seguintes:

- **criterion**: critério para medir a qualidade da divisão (gini, entropy)
- **max_depth:** profundidade máxima da árvore
- **max_features:** número máximo de features consideradas em cada divisão
- **min_samples_split:** número mínimo de amostras para dividir um nó.
- **min_samples_leaf:** número mínimo de amostras em uma folha

Decision Tree											
Prep	Parametros					f1 macro cross	Classification report			Control	kaggle f1
	criterion	max_depth	max_features	min_samples_leaf	min_samples_split		f1 accuracy	f1 macro	f1 weighted		
1	gini	10	log2	2	5	0,355277481	0,31	0,23	0,3	0,094	0,2684
2b	gini	10	-	2	2	0,35443128	0,41	0,32	0,42	0.1851	0,2882
3	gini	10	log2	4	2	0,338278735	0,34	0,24	0,3	0,075	0,2301
5	entropy	15	log2	2	10	0,331016509	0,36	0,28	0,35	0.072	0,1964

Figura 3: Resultados obtidos com Decision Tree

Analisando os resultados obtidos com este algoritmo, podemos verificar que encontramos bons resultados no treino, mas não excecionais. Observamos que no Kaggle obtivemos resultados bastantes baixos, o que pode indicar que o modelo está a sofrer overfitting nos dados de treino.

No entanto a análise do dataset de controlo revelou valores baixos, sugerindo que o modelo pode não estar a sofrer de overfitting, mas sim que a pontuação pública do Kaggle pode não refletir o verdadeiro desempenho do modelo nos dados privados.

6.2 Random Forest

O Random Forest é um algoritmo de ensemble que constrói múltiplas árvores de decisão e combina os seus resultados para melhorar a precisão e reduzir o overfitting. Os parâmetros explorados incluem:

- **n_estimators:** número de árvores na floresta
- **max_depth:** profundidade máxima das árvores
- **min_samples_split:** número mínimo de amostras para dividir um nó
- **min_samples_leaf:** número mínimo de amostras por folha
- **max_features:** número máximo de variáveis consideradas em cada divisão

Random Forest											
Prep	Parametros					f1 macro cross	Classification report			Control f1	kaggle result
	n_estimators	max_depth	min_samples_split	min_samples_leaf	max_features		f1 - accuracy	f1-macro	f1-weighted		
1	150	5	2	2	log2	0,3587	0,44	0,32	0,4	-	0,28242
2	150	10	2	2	log2	0,3387	0,48	0,35	0,43	-	0,35636
2	150	5	2	2	log2	0,3605	0,44	0,32	0,4	-	0,28242
2	100	10	2	2	log2	0,3358	0,51	0,37	0,46	-	0,33662
2	150	15	2	2	log2	0,3294	0,43	0,29	0,37	-	0,28671
2	200	5	2	2	log2	0,3460	0,44	0,32	0,4	0,133	0,32623
2b	50	15	10	2	sqrt	0,3694	0,44	0,3	0,39	-	0,31521
2b	50	10	10	2	sqrt	0,3495	0,46	0,32	0,41	-	0,29603
3	50	10	2	2	log2	0,3379	0,41	0,29	0,37	0,1745	0,44185
3	50	7	2	5	log2	0,3425	0,39	0,28	0,35	0,1583	0,38853
3	100	10	5	2	log2	0,3362	0,36	0,26	0,34	0,1789	0,4745
3 SMOTE	150	5	15	5	sqrt	0,6797	0,41	0,3	0,38	0,1057	0,3562
4	50	7	2	5	log2	0,3394	-	-	-	0,1286	0,38853
5	50	7	2	2	sqrt	0,3361	-	-	-	0,1276	-
5	150	10	5	1	sqrt	0,3393	0,44	0,33	0,41	-	0,35333
5 SMOTE	150	10	2	1	sqrt	0,7310	0,74	0,73	0,73	-	0,30026

Figura 4: Resultados obtidos com Random Forest

Este algoritmo apresentou um desempenho geral superior ao da Decision Tree, tanto no treino quanto no Kaggle.

Foi possível verificar que preparações simples, como a remoção de colunas categóricas (Preparação 3) ou a conversão destas para valores numéricos (Preparação 5), combinadas com normalização, resultaram em melhores scores no Kaggle. No entanto, preparações mais complexas, como as que removeram colunas altamente correlacionadas (Preparações 1, 2 e 2b), apresentaram melhores resultados no treino.

Apesar de as preparações mais simples mostrarem melhores resultados no Kaggle, isto pode ser apenas uma indicação de que esses ganhos ocorreram no score público, mas não necessariamente no score privado. Por exemplo, na Preparação 3, verificou-se que, no Kaggle, duas submissões apresentaram scores de 0,44185 e 0,4745 (no público), mas localmente os mesmos modelos obtiveram apenas 0,33, o que sugere overfitting. Isto também pode ser explicado pelo facto de termos definido um valor considerável para max_depth (10), o que pode ter levado o algoritmo a um sobreajuste aos dados de treino.

Além disso, neste algoritmo foi utilizado o SMOTE para balancear os dados. No entanto, os resultados mostraram scores muito elevados no treino (0,67 e 0,73), enquanto no Kaggle os scores foram consideravelmente mais baixos.

6.3 Support Vector Machine

O Support Vector Machine (SVM) é um algoritmo de classificação que tenta encontrar o hiperplano ideal para separar as diferentes classes de dados no espaço multidimensional. Os parâmetros utilizados foram:

- **C:** parâmetro de regularização;
- **class_weight:** pesos das classes para lidar com desbalanceamento;
- **degree:** grau do polinómio (quando usado o kernel polinomial);
- **gamma:** coeficiente do kernel
- **kernel:** tipo de kernel utilizado (linear, polinomial, RBF, sigmoid).

SVM											
Prep	Parametros					f1_macro cross	Classification report			Control	kaggle result
	C	class_weight	degree	gamma	kernel		f1 - accuracy	f1-macro	f1-weighted		
3	0,1	None	3	scale	poly	0,343764	0,43	0,3	0,38	0,0979	0,29047
2	10	balanced	3	auto	rbf	0,343457	0,39	0,29	0,37	0,077	0,31492
2b	0,1	balanced	3	scale	linear	0,335969	0,46	0,34	0,43	0,0715	0,15285
2b	100	balanced	2	scale	sigmoid	0,35644	0,48	0,34	0,43	0,1168	0,18241
2	10	balanced	2	auto	rbf	0,343457	0,39	0,29	0,37	0,077	0,31492

Figura 5: Resultados obtidos com Support Vector Machine

Assim como nas árvores de decisão, os resultados foram bons no treino, mas fracos no Kaggle. No entanto, a análise do dataset de controlo revelou valores reduzidos, sugerindo que o mau desempenho pode ocorrer apenas na partição pública e não na privada.

6.4 Gradient Boost

O Gradient Boost é outro algoritmo de ensemble que combina várias árvores de decisão sequencialmente, atribuindo maior peso aos erros cometidos nas iterações anteriores. Os principais parâmetros explorados foram:

- **learning_rate:** taxa de aprendizagem;
- **max_depth:** profundidade máxima das árvores;
- **min_samples_leaf:** número mínimo de amostras por folha;
- **min_samples_split:** número mínimo de amostras para dividir um nó;
- **n_estimators:** número de árvores no ensemble.

Gradient Boost											
Prep	Parametros					f1_macro cross	Classification report			Control	kaggle result
	lr	max_depth	min_samples_leaf	min_samples_split	n_estimators		f1 accuracy	f1 macro	f1 weighted		
3	0,1	5	2	2	50	0,36294061	0,43	0,32	0,41		0,2447
5	0,1	5	2	2	50	0,36514205	0,38	0,28	0,37	0,0296468	0,16634
3	1	10	2	5	100	0,32200367	0,34	0,26	0,34	0,1918606	0,35721
3	1	10	2	5	150	0,32200367	0,34	0,26	0,34	-	0,35721
3	1	10	2	10	150	0,36738451	0,39	0,3	0,38	-	0,29319
3	2	10	2	5	100	0,30977532	0,37	0,27	0,35	-	0,22394
1	1	10	2	5	100	0,3123299	0,41	0,3	0,38	-	0,30714
1	1	10	2	5	50	0,3123299	0,41	0,3	0,38	-	0,30714
1	0,1	10	2	5	50	0,3113906	0,46	0,37	0,45	-	0,3037
1	0,1	5	10	10	100	0,36127358	0,46	0,37	0,45	0,2062302	0,2833
3	0,1	10	2	5	100	0,35957027	0,43	0,33	0,42	-	0,28634
3 Random OverSampling	0,1	5	5	5	100	0,76197392	0,71	0,71	0,71	-	0,3318
3 Random OverSampling	1	10	2	5	100	0,7434628	0,68	0,66	0,66	-	0,23931
2	0,1	5	10	5	100	0,35423954	0,43	0,34	0,42	-	0,25695
3	0,1	10	2	5	300	0,36143755	0,43	0,33	0,41	-	0,31777
2b	0,1	10	10	5	50	0,34374229	0,52	0,41	0,51	0,1568479	0,38431
2b	0,1	5	5	5	300	0,33202252	0,59	0,46	0,57	0,2466501	0,32161
2b	0,1	5	10	5	50	0,3119579	0,51	0,39	0,5	0,2179152	0,34812
3	0,1	5	2	5	100	0,37498616	0,44	0,35	0,44	-	0,27251
3	0,3	5	5	10	100	0,37952178	0,43	0,33	0,42	0,109891	0,27743
2	0,1	3	5	5	150	0,38802471	0,52	0,42	0,51	0,1878558	0,2133

Figura 6: Resultados obtidos Gradient Boost

O Gradient Boosting destacou-se como um dos melhores algoritmos no treino, obtendo scores bastante elevados nesta fase. Além disso, apresentou alguns bons resultados no Kaggle, particularmente na pontuação pública.

Apesar do bom desempenho no treino e na pontuação pública do Kaggle, este algoritmo apresentou um F1-score elevado no conjunto de controlo. Isto pode indicar uma falta de generalização, sugerindo que o modelo pode estar ajustado para a partição pública do Kaggle, mas menos eficaz nos dados privados.

Foi utilizada a técnica de Random Oversampling para equilibrar as classes no dataset. Contudo, observou-se que, de forma semelhante ao SMOTE, esta abordagem resultou em scores F1 muito altos no treino, mas significativamente mais baixos no conjunto de teste. Isto reforça a ideia de sobreajuste ao treino.

Verificamos que learning rates menores, como 0.1, proporcionaram melhores resultados, uma vez que permitiram ao modelo fazer ajustes mais suaves e evitar oscilações drásticas no erro, e que o aumento

do valor de min_samples_leaf ajudou a melhorar a generalização do modelo, reduzindo o risco de overfitting ao exigir mais amostras por folha e simplificar as árvores.

6.5 XGB

O XGBoost (Extreme Gradient Boosting) é um algoritmo de ensemble baseado em árvores de decisão que utiliza a técnica de boosting para criar modelos robustos e de alta performance. Ele constrói sequencialmente árvores de decisão, otimizando os erros cometidos pelos modelos anteriores. Este algoritmo é conhecido pela sua eficiência e capacidade de lidar com datasets de grandes dimensões e com características desbalanceadas.

Os principais parâmetros ajustados foram:

- **colsample_bytree:** fração de variáveis utilizadas para construir cada árvore, permitindo regularizar o modelo;
- **gamma:** valor mínimo de redução na função de perda para permitir uma nova divisão de nó;
- **learning_rate:** taxa de aprendizagem que controla a contribuição de cada árvore para o modelo final;
- **max_depth:** profundidade máxima das árvores, controlando o grau de complexidade do modelo;
- **min_child_weight:** peso mínimo exigido para que um nó-folha seja dividido, ajudando a evitar overfitting;
- **n_estimators:** número de árvores no ensemble;
- **subsample:** fração de amostras utilizadas para treinar cada árvore, melhorando a generalização do modelo.

XGB													
Prep	Parametros							f1 macro cross	Classification report			Control	kaggle result
	colsample_bytree	gamma	learning_rate	max_depth	min_child_weight	n_estimators	subsample		f1 - accuracy	f1-macro	f1-weighted		
3	0,6	0,1	0,3	3	1	50	0,8	0,35672237	0,43	0,33	0,41	0,19	0,33351
2b	0,6	0,2	0,3	5	1	100	1	0,36102893	0,46	0,35	0,44	0,163	0,3178
3	0,6	0,1	0,1	3	1	50	0,8	0,31325745	0,51	0,39	0,49	0,16	0,46093
5	0,6	0,1	0,1	3	1	50	0,8	0,31338328	0,39	0,27	0,35	0,18	0,35298
2b	0,6	0,1	0,1	3	1	50	0,8	0,33426721	0,38	0,27	0,35	0,149	0,33071
2b	0,6	0,2	0,4	7	1	100	1	0,3724535	0,49	0,37	0,47	0,176	0,25333

Figura 7: Resultados obtidos com XGB

Analisando os resultados obtidos, foi possível observar que as preparações mais simples tiveram um desempenho superior no Kaggle (score público), enquanto as preparações mais complexas apresentaram resultados melhores nos dados de treino e de controlo.

No entanto, os scores elevados no controlo indicam que os modelos podem não estar a generalizar bem para novos dados, o que reforça a necessidade de ajustes mais criteriosos nos hiperparâmetros para garantir maior robustez e evitar overfitting.

6.6 Bagging

O Bagging (Bootstrap Aggregating) é um algoritmo de ensemble que combina várias instâncias de modelos base para melhorar a robustez e reduzir o overfitting. Os parâmetros ajustados foram:

- **n_estimators:** número de estimadores;
- **max_samples:** número máximo de amostras por estimador;
- **max_features:** número máximo de características consideradas;
- **bootstrap:** se as amostras são geradas com reposição;
- **bootstrap_features:** se as características são geradas com reposição;
- **estimatormax_depth:** profundidade máxima do modelo base;
- **estimatormin_samples_split:** número mínimo de amostras para dividir um nó no modelo base;
- **estimatormin_samples_leaf:** número mínimo de amostras em cada folha no modelo base.

Bagging														
Prep	Parametros								f1_macro cross	Classification report			control	kaggle
	n_estimators	max_samples	max_features	bootstrap	bootstrap_features	estimator__max_depth	estimator__min_samples_split	estimator__min_samples_leaf		f1 accuracy	f1 macro	f1 weighted		
2b	100	1	0.7	FALSO	FALSO	5	5	2	0,344236318	0,39	0,28	0,37	0,1494	0,17023
3	50	0.7	1	FALSO	FALSO	10	2	2	0,345708146	0,39	0,28	0,36	0,1	0,41452

Figura 8: Resultados obtidos com Bagging

Embora tenhamos realizado menos testes com este algoritmo, verificou-se novamente a tendência de que as preparações mais simples têm melhor desempenho no Kaggle, enquanto as mais complexas se destacam nos dados de treino.

6.7 MaxVoting

O Max Voting combina os resultados de vários algoritmos, escolhendo a classe com maior número de votos como a predição final. Neste caso, foram combinados os algoritmos Random Forest, SVM e Gradient Boosting. Este comportamento pode ser explicado pela diversidade entre os algoritmos usados na votação, o que reforça a necessidade de maior análise na escolha dos modelos a combinar.

Max Voting												
Prep	Parametros						f1 macro cross	Classification report			Control	kaggle
	rf__estimators	rf__max_depth	svc__C	svc__kernel	gb__n_estimatos	gb__lr		f1 accuracy	f1 macro	f1 weighted		
3	100	5	0,1	linear	50	0,1	0,35929109	0,41	0,29	0,37	0,1216	0,3367
2	100	3	0,1	linear	50	0,1	0,36120702	0,44	0,33	0,42	0,2059	0,3231
2b	100	3	0,1	linear	50	0,3	0,35747278	0,48	0,35	0,44	0,224	0,238

Figura 9: Resultados obtidos com MaxVoting

Os resultados mostraram-se promissores no treino, mas os scores no Kaggle foram baixos. Nos dados de controlo, o score foi superior à pure chance de 0.2, sugerindo que o modelo pode não estar a generalizar adequadamente.

6.8 Stacking

O Stacking é uma técnica de ensemble que combina múltiplos algoritmos (neste caso, SVC, Random Forest e Gradient Boosting) e utiliza um estimador final para fazer a predição (foi utilizada uma regressão logística como estimador final).

Stacking																		
Prep	SVC				RF					GB					LR	f1 cross	control	kaggle
	C	kernel	degree	gamma	estimators	max_depth	min_samples_split	min_samples_leaf	max_features	estimators	lr	max_depth	min_samples_split	min_samples_leaf	final_estimator__C			
3	0,1	poly	3	scale	50	10	2	2	log2	50	0,1	10	5	10	1	0,333	0,1416	
3	0,1	poly	3	scale	100	7	2	5	log2	100	0,1	5	5	2	1	0,353	0,1624	0,3964
2	10	-	-	-	100	10	-	-	log2	100	0,1	10	-	-	-	0,343	-	0.38476

Figura 10: Resultados obtidos com Stacking

No geral, este algoritmo apresentou bons resultados, particularmente no Kaggle, onde superou o desempenho obtido localmente. No entanto preparações que reduziram bastante o número de features, como o uso do PCA, deram um pior resultado no kaggle.

7 Análise dos Resultados

Fazendo uma análise global dos resultados obtidos em todos os modelos, podemos identificar padrões claros sobre a eficácia das diferentes abordagens testadas:

Impacto das Preparações no Desempenho:

- Preparações simples (ex.: remoção de colunas categóricas e normalização com Min-Max) demonstraram melhores resultados na partição pública do Kaggle. Estas preparações, por não alterarem drasticamente o número de variáveis, permitiram que os modelos captassem mais informações do dataset original, o que aparentemente resultou em maior adaptação aos dados do Kaggle.
- Por outro lado, preparações mais complexas (ex.: eliminação de colunas altamente correlacionadas ou aplicação de técnicas de redução de dimensionalidade, como PCA) apresentaram resultados superiores nos dados de treino. Isso pode indicar que estas técnicas ajustaram o modelo de forma mais detalhada ao treino, mas não necessariamente ajudaram na generalização.

Técnicas de Balanceamento:

- Técnicas como SMOTE e Random Oversampling, usadas para resolver o desbalanceamento da variável dependente, contribuíram para scores elevados no treino. Contudo, foi notado que estas abordagens frequentemente resultaram em overfitting, pois o desempenho no conjunto de teste foi significativamente inferior. Isto demonstra que, embora estas técnicas possam ser úteis para lidar com datasets desbalanceados, podem piorar a generalização.

Para selecionar o melhor modelo entre todos os testados, foi priorizado aquele que apresentou um equilíbrio entre bom desempenho no treino e na partição pública do Kaggle. O modelo escolhido foi um Gradient Boosting treinado com a preparação 2b, que consistia em:
- Remoção de colunas categóricas;
- Remoção de colunas com correlação superior a 0.8;
- Normalização utilizando Min-Max Scaling.

Este modelo demonstrou os seguintes resultados:
- F1 macro no treino com cross validation: 0.34
- F1 macro no treino : 0.41
- F1 macro no Kaggle (público): 0.38

Apesar do bom desempenho no treino e na partição pública do Kaggle, o score obtido na partição privada do Kaggle foi de 0.34, ligeiramente abaixo do esperado. Um fator que pode explicar o desempenho abaixo do esperado no Kaggle privado é o F1-score de 0.15 obtido no conjunto de controlo. Este valor, ainda que inferior a 0.2 (valor da pure chance 1/5 = 0.2), sugere que o modelo teve dificuldade em generalizar para novos dados.

Após análise dos resultados na partição privada do Kaggle, constatou-se que modelos inicialmente considerados fracos devido ao baixo desempenho na partição pública do Kaggle se destacaram na partição privada.

O caso mais evidente foi uma Árvore de Decisão, que:

- Obteve F1 macro no Kaggle (público): 0.26838;
- Alcançou F1 macro no Kaggle (privado): 0.45469;
- Apresentou um F1 macro no treino com cross-validation: 0.355.

Este desempenho surpreendentemente bom na partição privada revela que, embora este modelo tenha tido resultados modestos no treino e na partição pública, ele foi capaz de generalizar melhor para a partição privada.

8 Reflexão e pontos a melhorar

Fazendo uma reflexão sobre o que foi o nosso trabalho, podemos afirmar que os objetivos estabelecidos foram alcançados com sucesso. Neste trabalho usamos vários métodos de preparação de dados de forma a compreender como estes influenciam o desempenho dos modelos. Além disso exploramos uma ampla gama de algoritmos de aprendizagem automática, identificando os seus pontos fortes e limitações em diferentes contextos.

Durante o trabalho, verificamos que o score público do Kaggle tem relevância limitada para avaliar o verdadeiro desempenho do modelo, uma vez que pode estar enviesado devido à distribuição desbalanceada das classes e ao número reduzido de instâncias por classe na partição pública. Assim, concluímos que o resultado no treino, sobretudo utilizando cross-validation, é crucial para avaliar o desempenho consistente do modelo, assim como o dataset de controlo é essencial para verificar a capacidade do modelo de generalizar para novos dados, além de assegurar que não há vazamento de informação.

Identificamos também algumas áreas que podem ser melhoradas em trabalhos futuros como a implementação de novos algoritmos, tais como redes neuronais para explorar a sua capacidade de aprender representações mais complexas. Outro ponto seria a exploração de algoritmos de aprendizagem não supervisionada, como clustering (K-Means, DBSCAN) ou técnicas de dimensionalidade para identificar padrões nos dados e melhorar a sua preparação.

9 Conclusão

Concluímos que o trabalho desenvolvido foi, em geral, satisfatório, tendo alcançado os principais objetivos definidos para este projeto. Através de uma análise detalhada dos dados e da aplicação de técnicas de aprendizagem automática, conseguimos compreender melhor o problema em questão e apresentar resultados coerentes e fundamentados.

Durante o desenvolvimento, foi possível perceber a importância de testar várias preparações de dados e algoritmos, especialmente num dataset com um número reduzido de entradas, onde é particularmente desafiador alcançar resultados consistentes e evitar problemas como overfitting ou falta de generalização.