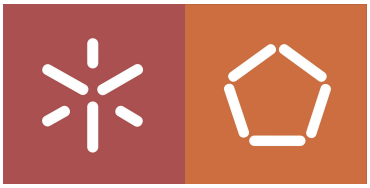

Universidade do Minho
Departamento de Informática

# Aprendizagem Profunda

Ano Letivo 2024/2025

# Relatório do Trabalho Prático

Lucas Oliveira - PG57886

João Barroso - PG57554

Maurício Pereira - PG55984

Rafael Gomes - PG56000

# Índice

# 1. Introdução

Este relatório foi elaborado no contexto do trabalho prático da unidade curricular de Aprendizagem Profunda, com o propósito de investigar a criação de modelos de *Deep Learning* recorrendo às técnicas estudadas durante o semestre.

O trabalho envolve a construção e preparação de datasets em língua inglesa, a experimentação com diferentes tipos de modelos e a avaliação rigorosa dos resultados obtidos.

Todo o conteúdo realizado para este trabalho prático pode ser encontrado no repositório https://github.com/RafaGomes1/AP_2025

## 1.1. Contexto

O principal objetivo deste trabalho é desenvolver modelos capazes de distinguir entre textos gerados por sistemas de Inteligência Artificial (*IA*) e textos escritos por seres humanos. Para isso, são exploradas diferentes abordagens e arquiteturas de *Machine Learning* e *Deep Learning*, recorrendo tanto à implementação manual dos modelos como à utilização de *frameworks* como o *TensorFlow*.

# 2. Metodologia para a Criação do *Dataset*

Para o desenvolvimento deste trabalho, foi adotada a seguinte metodologia para a construção do nosso *dataset*:

- **Procura de *Datasets*:** inicialmente, além dos *datasets* indicados no enunciado pela equipa docente, foi realizada uma pesquisa adicional em plataformas como o *Kaggle* e o *Hugging Face*, com o objetivo de encontrar conjuntos de dados complementares.
- **Análise dos *Datasets*:** após a recolha dos *datasets*, procedeu-se à análise das suas estruturas com o objetivo de verificar quais se adequavam aos requisitos definidos para a realização do trabalho prático.
- **Escolha dos *Datasets*:** após a análise, selecionámos os *datasets* que consideramos mais adequados para o treino dos nossos modelos e para a realização das previsões finais.
- **Avaliação dos *Datasets* escolhidos:** com os *datasets* selecionados, avaliamos o desempenho dos modelos durante o treino e nas previsões finais. Concluímos que os resultados obtidos foram insatisfatórios, uma vez que os modelos tendiam a prever apenas uma das classes — ou apenas *Human* ou apenas *IA*, demonstrando um claro desbalanceamento na classificação.
- **Junção de Dados de *Datasets*:** perante os resultados obtidos, procedemos à reestruturação dos nossos *datasets*, optando por combinar dados provenientes de diferentes fontes. Esta abordagem visou aumentar a diversidade dos exemplos disponíveis, promovendo uma melhor generalização e desempenho dos modelos durante o treino.

# 3. Modelos de Raiz

Com o objetivo de estabelecer linhas de base (*baselines*) de desempenho, foram desenvolvidos modelos de raiz recorrendo a técnicas tradicionais de *Machine Learning*. Os métodos implementados nessas submissões iniciais serviram de parâmetro comparativo para avaliar melhorias proporcionadas mais tarde pelos modelos *Deep Learning*, descritos nas secções seguintes.

## 3.1. Submissão 1 (*S1*)

Na primeira submissão (*S1*), procedeu-se à implementação de um *pipeline* que incluiu:

**1. Limpeza e pré-processamento dos dados:**

- Remoção de caracteres especiais, pontuação e espaços em excesso.
- Conversão de todo o texto para minúsculas (*lowercasing*).
- *Tokenização* (separação em palavras/pedaços).
- Remoção de *stopwords* (palavras de pouco valor semântico, como *"o"*, *"a"*, *"de"*, *etc*).

**2. Transformação (*Vectorization*):**

- Aplicação do *TF-IDF* (*Term Frequency - Inverse Document Frequency*) para representar cada documento de forma numérica, capturando a relevância de cada termo no contexto de todos os documentos do conjunto.

**3. Modelos de *Machine Learning*:**

- *Naive Bayes* (*MultinomialNB*)
- *Logistic Regression*
- *Random Forest*
- *SVM* (*Support Vector Machines*)

Cada modelo foi treinado numa partição de treino e avaliado sobre um conjunto de validação. Além disso, realizou-se uma análise comparativa usando métricas de desempenho clássicas (*accuracy*, *precision*, *recall* e *F1-score*).

## 3.2. Submissão 1 (*S2*)

A segunda submissão (*S2*) procurou trazer variações ao *pipeline* acima descrito, mas mantendo a mesma lógica geral de modelos de *Machine Learning*. As principais diferenças residem em:

**1. Ajuste de Hiperparâmetros (*Hyperparameter Tuning*):**

- Foram testados valores diferentes de *C* (parâmetro de regularização) para *SVM* e *Logistic Regression*.
- Ajustou-se o número de *estimators* no *Random Forest* e o critério de divisão (*Gini vs Entropy*).
- No *Naive Bayes*, foi alterada a forma de suavização (*Laplace vs. Lidstone*).

**2. Estratégias de Validação Diferentes:**

- Alguns testes foram feitos recorrendo a *cross-validation* (*hold-out vs. K-fold de 5 ou 10*).
- Em certos casos, introduziu-se estratificação para garantir a mesma proporção de classes em cada *fold*.

**3. Revisão do Pré-Processamento:**

- Experimentaram-se diferentes listas de *stopwords* (mais abrangentes ou mais específicas).

- Testou-se a aplicação de *lemmatization* ou *stemming*, embora em alguns cenários isso tenha introduzido pouco ganho efetivo.

# 4. Modelos *Tensorflow*

Com o objetivo de melhorar significativamente os resultados obtidos nas submissões anteriores com modelos de *Machine Learning* clássicos, implementados manualmente, nesta secção iremos explicar o uso da *framework TensorFlow*, que nos permitiu o desenvolvimento de modelos de *Deep Learning* de forma eficiente e flexível.

## 4.1. Submissão 2 (*S1*)

Nesta submissão, foi desenvolvido um modelo *Deep Learning* utilizando a *API Keras* do *TensorFlow*. E consistiu das seguintes etapas:

**Pré-Processamento**

os textos foram convertidos para minúsculas, *tokenizados* e transformados em sequências de inteiros. Para a tokenização, utilizámos o *Tokenizer* do *Keras* com um vocabulário limitado a **20.000 palavras** e um token especial para palavras fora do vocabulário **(<OOV>)**. As sequências foram depois padronizadas com *pad_sequences*, utilizando um comprimento fixo de 500 tokens, garantindo assim uma entrada uniforme para o modelo.

**Arquitetura do Modelo**

O modelo desenvolvido é do tipo *Sequential*, com a seguinte composição:

- ***Embedding Layer***: foi utilizada uma camada de *embedding* com dimensão de *128*, convertendo os *tokens* em vetores densos.
- **Camadas Ocultas**:
  - Uma *Flatten* para transformar os embeddings num vetor unidimensional.
  - Uma *Dense* com **128 unidades** e ativação **ReLU**.
  - *Dropout* de **0.4** para mitigar overfitting.
  - Uma *Dense* com **64 unidades** e ativação ***ReLU***
- **Camada de Saída**: uma *Dense* com ativação **sigmoide**, apropriada para classificação binária (*IA vs Humano*).
- **Treino**: o modelo foi treinado com o otimizador ***Adam***, a função de perda ***binary_crossentropy***, durante 15 épocas com ***cross-validation***, com o tamanho do batch 64.

## 4.2. Submissão 2 (*S2*)

Nesta submissão, explorou-se uma arquitetura similar à da secção anterior, mas com algumas modificações nos hiperparâmetros:

- O número de épocas diminuiu para 10.

## 4.3. Outros Modelos

Para além dos modelos falados na secção *4.1* e *4.2*, foram explorados outras abordagens experimentais com o intuito de investigar alternativas promissoras.

### 4.3.1. Modelos adicionais em *TensorFlow*

Estas experiências serviram também para melhor compreensão do impacto de diferentes arquiteturas, pré-processamentos e frameworks.

#### 4.3.1.1. CNN + Bidirectional LSTM + DNN com Embeddings GloVe

Este modelo experimental combina convoluções, redes recorrentes bidireccionais e camadas densas (DNN), tirando partido de embeddings pré-treinados GloVe para capturar semântica linguística:

- **Embeddings**: Utilização de vetores *GloVe* (*Global Vectors for Word Representation*), que fornecem representações densas semânticas para as palavras, carregadas a partir de ficheiros externos.
- **Arquitetura**:
  - Camada de **Embedding** inicializada com pesos *GloVe*.
  - Camada **Convolucional 1D (CNN)** para extração de padrões locais.
  - Camada **Bidirectional LSTM** para capturar dependências contextuais na sequência (tanto à frente como para trás).
  - Camadas ***Dense* (DNN)** para classificação final com ativação *sigmoid*.
- **Regularização**: Camadas de *Dropout* aplicadas entre as camadas para mitigar overfitting.

Este modelo mostrou-se eficaz em capturar tanto padrões locais (via CNN) como dependências a longo prazo (via LSTM), mas apresentava maior custo computacional.

#### 4.3.1.2. Classificador Keras com Embedding, GlobalAvgPooling1D e regularização extensiva

Neste modelo, foi adotada uma arquitetura mais leve, porém robusta, modularizada em diferentes scripts Python, otimizando a reusabilidade e manutenção.

- **Tokenização**: Feita com *Tokenizer Keras* com *oov_token*, vocabulário limitado e padding fixo (500 tokens).
- **Arquitetura**:
  - Camada *Embedding* para mapear tokens a vetores densos.
  - Camada *GlobalAveragePooling1D* para reduzir a dimensionalidade.
  - Camada *Dense* com ativação *LeakyReLU*.
  - Aplicação de **BatchNormalization** e ***Dropout*** para regularização.
  - Camada de saída *Dense* com ativação *sigmoid*.
- **Treino**:
  - Uso de **EarlyStopping** com base na perda de validação para evitar overfitting.

Este modelo destaca-se pela simplicidade e capacidade de generalização, sendo uma das soluções mais equilibradas entre performance e custo.

#### 4.3.1.3. Híbrido CNN + Bidirectional LSTM com GloVe + Regularização

Este modelo representa uma fusão entre as abordagens anteriores, combinando convoluções, LSTM bidirecional, e uso de embeddings GloVe.

- **Embeddings**: Vetores GloVe pré-treinados carregados manualmente.
- **Arquitetura**:
  - Camada *Embedding* com pesos GloVe congelados.
  - Camada *Conv1D* seguida de *MaxPooling*.
  - Camada *Bidirectional LSTM*.
  - Camadas *Dense* para classificação.
- **Regularização**:
  - *Dropout* após as camadas recorrentes e densas.
  - *BatchNormalization* para estabilizar o treino.

O desempenho deste modelo foi competitivo, mas o tempo de treino e a complexidade computacional tornaram-no menos prático para submissões rápidas.

Para além dos modelos mencionados, foram experimentadas variantes dos modelos principais com diferenças em:

- **Arquitetura**: inclusão de mais camadas densas, alteração da dimensão de *embeddings* e diferentes funções de ativação.
- **Regularização**: testes com valores distintos de ***dropout***, ***batch normalization*** e estratégias de **early stopping**.
- **Estratégias de otimização**: foram testados otimizadores como ***RMSprop***, ***SGD*** e variantes do *Adam* (como ***Adamx***), com diferentes valores de *learning_rate*.

#### 4.3.2. Exploração com Transformers

Realizamos também uma primeira abordagem com modelos pré-treinados da biblioteca *Hugging Face transformers*, nomeadamente com o modelo ***DistilBert*** para classificação de texto, em que o pipeline envolveu:

- **Tokenização** com *DistilBertTokenizerFast*.
- **Uso do modelo** *TDFistilBertForSequenceClassification*
- **Treino com** *TFTrainer* e respetivos datasets em formato *TFDataset*

Mas apesar da complexidade e maior capacidade representacional deste modelo, a *accuracy* obtida não ultrapassou os *76%*, o que possivelmente pode ser devido à necessidade de maior volume de dados ou até mesmo um *fine-tuning* mais extensivo.
Ao explorar estes modelos adicionais, reforçou ainda mais a decisão de manter os modelos falados na secção *4.1* e *4.2* como os principais para submissão, pois atingiram um melhor equilíbrio entre desempenho, simplicidade e interpretabilidade.

# 5. Modelos *Claude*

Além dos modelos de raiz e dos modelos construídos em *TensorFlow*, foi também explorada a utilização de Modelos *Claude*, desenvolvidos pela *Anthropic*. Estes modelos baseiam-se em grandes modelos de linguagem (*LLMs*) e foram acedidos através da *API* disponibilizada pela empresa, permitindo gerar e classificar texto de forma altamente sofisticada.

## 5.1. Integração via *API* da *Anthropic*

Para a utilização dos modelos da *Anthropic* (*Claude*), foi necessário criar credenciais de acesso e efetuar chamadas *REST* à *API*. O processo inclui:

**1. Configuração de Autenticação**

- Criação de uma *API Key* associada ao projeto.
- Armazenamento seguro desta chave em variáveis de ambiente para evitar exposição pública do *token*.

**2. Formato de Requisição**

- Envio de *prompts* de texto para o modelo, juntamente com parâmetros de configuração, tais como *temperature*, *max_tokens* e *stop sequences*.
- Receção da resposta contendo a predição/classificação ou o texto gerado pelo modelo.

**3. *Pipeline* de Classificação**

- Preparação dos exemplos de teste.
- Construção de um *prompt* específico, instruindo o modelo a determinar se o texto em análise foi escrito por um ser humano ou gerado por *IA*.

- Análise automática das respostas do *Claude* para aferir a correção em relação aos rótulos do conjunto de teste.

### 5.2. Modelos Avaliados

Ao longo dos testes, foram exploradas distintas versões do *Claude*:

- *claude-3-7-sonnet-20250219*
- *claude-3-5-sonnet-20241022*

(Outras versões como *Claude 3.5 Haiku*, *Claude 3 Haiku* ou *Claude 3 Opus* também foram experimentadas, mas com menor foco nos resultados finais.)

Cada variante difere em termos de tamanho do modelo, *training data cutoff*, estrutura interna e ajustes finos (*finetuning*) realizados pelos investigadores da *Anthropic*. Na prática, pequenas alterações no modelo podem impactar a capacidade de compreender e classificar textos de forma correta.

## 6. Resultados

### 6.1. Modelos Raiz

O melhor desempenho foi obtido com o *SVM*, que alcançou uma *accuracy* de *58%* no conjunto de teste.

Observou-se que a *Logistic Regression* e o *Random Forest* apresentaram resultados ligeiramente abaixo, enquanto o *Naive Bayes* mostrou melhor capacidade de generalização quando o conjunto de dados estava desequilibrado, mas ainda assim não superou o *SVM* em termos de *accuracy* global. O resultado de 58% de *accuracy* representou um valor razoável como ponto de partida, indicando que o modelo conseguia distinguir textos gerados por *IA* de textos humanos significativamente acima do aleatório.

Contudo, ainda há espaço para melhoria; o pipeline de pré-processamento, a seleção de hiperparâmetros e a estratégia de balanceamento de classes poderiam ser otimizados.

### 6.2. Modelos *Tensorflow*

Relativamente aos modelos que foram desenvolvidos com a *framework TensorFlow*, designados *S1* e *S2*. Em ambos os casos, construiu-se uma rede neuronal composta por camadas totalmente ligadas (*fully connected layers*), intercaladas com a função de ativação *Rectified Linear Unit* (*ReLU*) e camadas de *dropout* para prevenir *overfitting*. Utilizou-se o *Adaptive Moment Estimation* (*Adam*) como método de otimização e a função de perda *cross-entropy*.

Após o treino e a avaliação dos modelos num conjunto de teste, a *S1* e *S2* apresentaram ambas *79%* de *accuracy*. Este resultado supera significativamente os valores dos modelos de raiz, que se situavam na casa dos *50–60%*.

### 6.3. Modelos *Claude*

Em comparação com os modelos de raiz e com outras abordagens de redes neuronais tradicionais, os modelos *Claude* demonstraram um desempenho significativamente superior. Destacam-se:

***claude-3-7-sonnet-20250219:***

Obteve uma *accuracy* de *97%*, sendo o melhor resultado registado no nosso projeto.

***claude-3-5-sonnet-20241022:***

Apresentou uma *accuracy* de *94%*, também acima dos valores alcançados pelos restantes modelos testados.

Esses valores refletem o estado da arte atual em modelos de linguagem de larga escala, exibindo grande capacidade de identificar padrões subtis que indicam se um texto foi gerado por *IA* ou não.

**Generalização e Robustez**

- Ao analisarmos diferentes domínios textuais (desde textos curtos informais até redações mais longas e formais), verificou-se que o *Claude* manteve elevada consistência na classificação, mesmo quando confrontado com *prompt engineering* destinado a confundi-lo.

**Comparação com Modelos Anteriores**

- Os resultados superam facilmente os das soluções baseadas em métodos clássicos de *Machine Learning* e também mostram-se superiores aos alcançados com redes neuronais menos sofisticadas.
- A integração via *API* proporcionou facilidade de uso, permitindo testar rapidamente diferentes *endpoints* do *Claude* e ajustar parâmetros de inferência.

**Custo e Limitações**

- Apesar do alto desempenho, deve-se considerar o custo computacional e financeiro associado ao consumo da *API*, especialmente em cenários de processamento em larga escala.

A utilização dos modelos da *Anthropic (Claude)* demonstrou claramente a eficácia de *LLMs* no contexto de distinção entre textos gerados por *IA* e textos humanos, alcançando *accuracies* na ordem dos *94%* a *97%*. Tais resultados reforçam:

A viabilidade de soluções prontas baseadas em *APIs* comerciais para classificação de textos.

O poder de generalização dos *LLMs*, capazes de lidar com uma grande variedade de estilos e formatos de texto.

O potencial de adoção futura desses modelos em sistemas que exijam elevada fiabilidade na deteção de textos artificiais.

No geral, estas experiências evidenciam que *Claude* oferece atualmente o melhor desempenho dentro do nosso conjunto de modelos, constituindo uma referência de comparação para quaisquer evoluções posteriores na área de classificação de autoria de texto.

## 6.4. Tabela Geral dos Resultados Obtidos

| Modelo | Tipo de Arquitetura | Accuracy(%) | Observações Principais |
|---|---|---|---|
| S1 (Submissão 1) | SVM com TF-IDF | 58% | Melhor entre os modelos clássicos |
| S2 (Submissão 1) | SVM / RF / NB + tuning | 53% | Pequenos ganhos com tuning e validação cruzada |
| S1 (Submissão 2) | Embedding + Flatten + Dense + Dropout | 79% | Arquitetura simples e eficiente com boa generalização |
| S2 (Submissão 2) | Igual à S1 com menos épocas | 79% | Resultados semelhantes, possível overfitting mais controlado |
| Claude 3.5 / 3.7 (API) | LLM da Anthropic via API | 94–97% | Melhor performance; custo computacional/financeiro elevado |

A tabela acima permite observar claramente a evolução do desempenho dos modelos ao longo do projeto. Verifica-se uma melhoria significativa ao transitar de modelos clássicos de Machine Learning para redes neurais com TensorFlow, com um salto de aproximadamente 20 pontos percentuais na accuracy.
É importante realçar que a escolha do modelo ideal depende do contexto de aplicação. Enquanto Claude é claramente superior em termos de precisão, os modelos desenvolvidos localmente oferecem maior

controlo, interpretabilidade e escalabilidade, podendo ser mais apropriados em ambientes restritos ou com limitação de recursos.

# 7. Conclusão

O principal objetivo deste trabalho foi a distinção entre textos gerados por sistemas de *IA* e textos escritos por humanos, recorrendo a diferentes abordagens de *Machine Learning e Deep Learning*. Iniciou-se com métodos de raíz (baseados em *Machine Learning* tradicional), obtendo-se resultados na ordem dos *53–58%* de *accuracy*, o que serviu de linha de base para validar e comparar soluções mais avançadas.

De seguida, a adoção de redes neuronais simples em *TensorFlow* permitiu elevar a *accuracy* para *79%*, demonstrando o potencial de arquiteturas de *Deep Learning*, mesmo num estágio inicial de complexidade. Por fim, a utilização de grandes modelos de linguagem (*LLMs*) disponibilizados pela *API* da *Anthropic* (*Claude*) apresentou desempenhos notavelmente superiores, chegando a atingir *94%* e *97%* de *accuracy* nas versões testadas. Estes resultados evidenciam a evolução notável que modelos recentes de linguagem conseguem alcançar na tarefa de deteção de textos artificiais.

Em termos de impacto, os achados deste projeto confirmam a eficácia de métodos avançados de *Deep Learning* na identificação de padrões textuais gerados por *IA*, embora com custos computacionais e financeiros mais elevados. Além disso, torna-se claro que a engenharia do *prompt* e a qualidade dos dados de treino podem ser decisivas para tirar o máximo partido destes modelos. Para trabalho futuro, será relevante explorar arquiteturas híbridas, estratégias de *data augmentation* e técnicas de transferência de conhecimento para lidar com escalas maiores de dados, bem como consolidar a robustez dos sistemas perante novos estilos de geração textual.