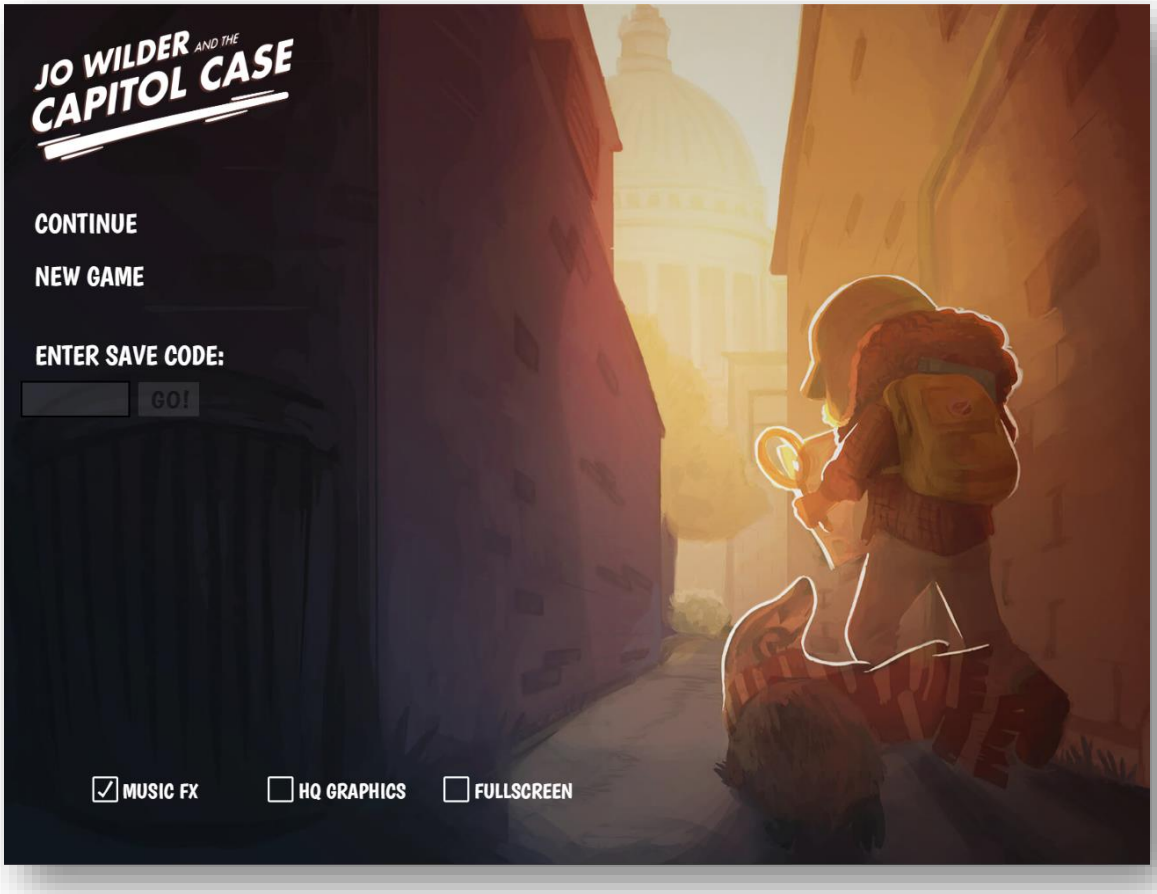
# Processamento De Big Data

## TEMA:

## PREDICT STUDENT PERFORMANCE FROM GAME PLAY




TRABALHO REALIZADO PELO GRUPO 12:
DIOGO FREITAS Nº104841 TURMA CDB1
JOÃO BOTAS Nº104782 TURMA CDB1

# Introdução

Neste trabalho serão abordados vários passos para se conseguir aplicar algoritmos de machine learning na base de dados Predict Student Performance From Game Play, sendo que esta possui informações de desempenho de estudantes, entre 8 e 12 anos, que jogaram o jogo Jo Wilder and the Capitol Case, um jogo de puzzles e mistérios sobre a história de *Wisconsin*. Assim, os jogadores vão poder aumentar o espírito crítico e aprender conceitos históricos sobre a cidade de *Wisconsin*.

Serão abordados vários tópicos, desde o entendimento do jogo e das respetivas variáveis recolhidas, passando pelo tratamento destas variáveis e o solucionamento de vários problemas que foram surgindo na realização deste trabalho, até à aplicação de algoritmos de *machine learning* que visam prever as respostas dos estudantes às questões do jogo.

Este trabalho tinha como objetivo compreender melhor as linguagens de programação *python* e *pyspark*, como também ter a oportunidade de testar e experimentar, pela primeira vez, a *cloud AWS* que facilitou o processo de aplicar os algoritmos de *machine learning*, com os dados por inteiro.

Por fim, serão extraídas conclusões não só sobre todos os processos de construção do trabalho, não deixando de fora os seus respetivos erros e problemas, como também serão comentados os resultados obtidos nos algoritmos, sendo analisado apenas o melhor deles, que estará sujeito a uma comparação dos resultados no AWS e nos nossos computadores pessoais.

# Problema em estudo

Neste projeto, a base de dados utilizada para estudo e previsão foi a Predict Student Performance From Game Play (*link* para os dados), que nos forneceu dados de jogadores, como a respetiva localização do mouse, se este estava a jogar em tela cheia, com música, etc…, das pessoas que jogaram o jogo Jo Wilder and the Capitol Case. O objetivo deste jogo é desvendar os mistérios e responder corretamente às questões, existindo um total de 18 perguntas. As questões são feitas no *notebook* da protagonista do jogo.

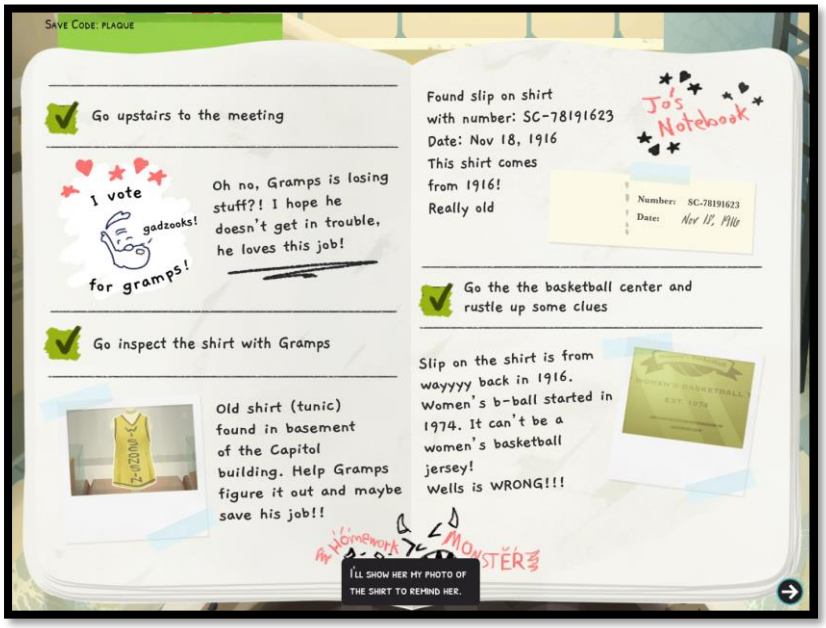



Figura 1 – Notebook da protagonista do jogo

Este trabalho tem como objetivo prever as respostas dos jogadores para cada questão, isto é, tentar prever se a questão foi bem, ou mal respondida. Ter em atenção que teremos sempre como argumento ***positive*** (classe de referência) as respostas dadas como corretas.

# Dados utilizados

Para a realização deste trabalho, o grupo teve acesso a 4 tabelas diferentes. As 2 maiores, que continham maior parte dos dados (estando estas intituladas de ***test.csv*** e ***train.csv***), o grupo decidiu juntá-las, criando uma tabela jogos, com a finalidade de apenas dividir em treino/teste na hora da realização dos modelos. Importante salientar que o ficheiro de teste tinha mais uma coluna que o ficheiro de treino, mas que foi retirada, visto que correspondia à mesma informação que outra coluna existente nos dois ficheiros (***level_group***, mas com *encoding*).

Já as respostas às perguntas de cada um dos jogadores, como também a identificação se esta resposta está correta ou incorreta, encontram-se na tabela ***train_labels.csv***. Ainda, era-nos dada uma tabela intitulada de ***sample_submission.csv*** que deveria ser a forma de como submeter as nossas respostas na competição, mas o grupo optou por não usar esta tabela. O que foi dito está descrito na imagem abaixo.

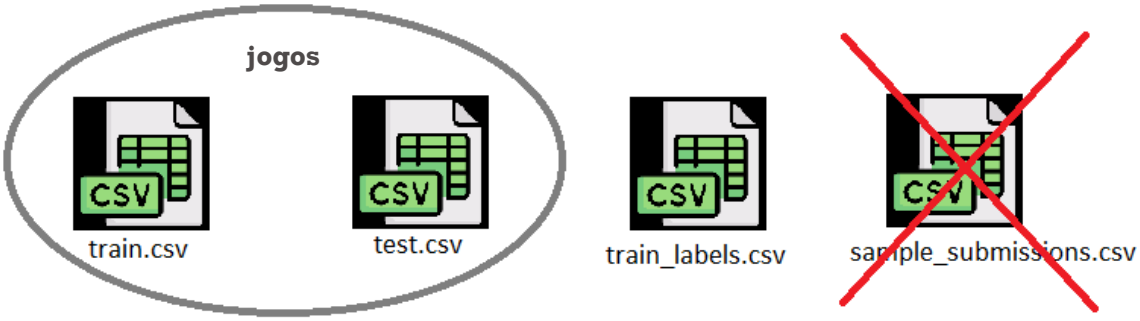



Figura 2 – As 4 tabelas da base de dados

No próprio site do *Kaggle* é referido o uso da biblioteca *Pandas* do *Python* para a realização deste concurso e, como o nosso grupo não usou a biblioteca *Pandas* diretamente, acreditamos que os nossos resultados/métodos, como também as nossas interpretações aos problemas possam ter sido um pouco diferentes.

# Problemas em relação aos dados

Logo após a instalação dos dados, o grupo deparou-se com vários problemas, sendo estes: "Em que níveis as perguntas são feitas?" e "Os dados e as respostas dadas às perguntas encontram-se em *Data Frames* diferentes". Para resolver estas questões, foi necessário ter algum conhecimento prévio do jogo, portanto, o grupo teve de jogar o jogo em estudo para poder começar a formular soluções.

Ao jogar, obtivemos as respostas que procurávamos como também obtivemos um melhor esclarecimento de várias outras, como o tipo de eventos (***event_name***), ou o nome das salas (***room_fqid***). De forma resumida, o jogo é dividido em 5 capítulos, em que só 3 deles têm as 18 perguntas, respondidas numa sala designada "capitólio", através de um ***notebook_click***, no diário de *Jolie*. Ou seja, a resposta é dada com base nas provas que a personagem principal encontra na cidade toda e respondida com cliques no *notebook* que evidenciem as pistas (uma resposta errada leva a uma fala da questionadora e novas tentativas de resposta até acerto).

Com estas ideias mais presentes, decidimos separar as 18 perguntas numa tabela, tal como a resposta considerada correta para cada uma delas. Foi destacado de cor diferente o agrupamento por fases (***level_group***) que ocorreu a resposta. Abaixo observa-se a tabela resultada desta análise:

Tabela 1 – As perguntas do jogo, com as respetivas respostas

| Nº da pergunta | Pergunta | Resposta |
|---|---|---|
| 1 | Which artifact? | jersey |
| 2 | How do you know? | slip on the shirt 1916 |
| 3 | How does that prove our shirt is not a jersey? | slip was before |
| 4 | So? What´s the story? What slip? | dry cleaner |
| 5 | Who told me it was a cleaning slip? | clothes expert |
| 6 | How did I find the right dry cleaner? | logbook |
| 7 | What did I find out from the logbook? | Theodora Youmans |
| 8 | How is Youmans famous? | suffragist |
| 9 | Where does the shirt come in? | marching for women´s rights |
| 10 | How do you know he was kidnapped? | scarf on the floor |
| 11 | What else did I find at the crime scene? | cup of coffee |
| 12 | How does a coffee cup prove Wells took Teddy? | spot at bean town |
| 13 | I think Wells is going to have Teddy stuffed and mounted. Why? | Taxidermist |
| 14 | Is the symbol a deer hoof? | wildlife rescue (not a deer track) |
| 15 | What does our flag stand for? | ecology flag |
| 16 | What does the symbol mean? | letter theta |
| 17 | What does the symbol mean? | Governor Nelson on the 1st earth day |
| 18 | Did you find any photos or the exhibit? | earth day flag on the capitol |

Assim, concluímos que as perguntas 1 a 3 pertenciam ao *level_group 0-4*, as perguntas 4 a 13 ao *level_group 5-12* e as restantes ao *level_group 13-22*, contribuindo a resposta ao primeiro dilema enunciado acima. Esta conclusão foi efetuada no ficheiro ***train_labels.csv*** com a criação de uma função (*generateLevelgroup*) que associava o número da pergunta com o seu respetivo nível. É necessário referir que apenas a primeira resposta à pergunta é guardada na base de dados, ou seja, se a pessoa errar à primeira tentativa, estamos a considerar uma resposta incorreta.

Nota: os outros dois capítulos em que não constam perguntas são o resgaste do *Teddy* e o pôr do sol fora da cidade.

# Feature engineering

Após esta fase de estudo dos dados, o grupo então teve de começar a preparar os dados para finalmente conseguir aplicar os modelos. Após várias pesquisas e várias tentativas de conseguir arranjar uma forma de conseguir aplicar modelos, o grupo chegou à conclusão que seria necessário criar uma tabela que possuísse todos os dados de resposta às perguntas apenas (resposta ao segundo questionamento).

Primeiramente, o grupo começou por criar uma tabela que apenas continha as variáveis ***session_id, level_group, question*** e ***correct.*** De seguida, foi decidido criar uma variável com a

probabilidade de acertar cada pergunta (de forma a arranjar alguma forma de as diferenciar), sendo esta intitulada de ***ProbCorrect***. Após isso, foi realizado um "***right join*"** com dados descritivos da tabela **jogos**, onde tivemos por base (colunas em comum) a ***session_id*** e o ***level_group***, tendo sido necessário fazer a média das várias variáveis numéricas, exceto o **elapsed_time**, que dizia o tempo total da sessão de jogo, e o ***index***, que continha o número de cliques do jogador. Em ambas decidiu-se escolher apenas o valor máximo. Já na lista dos vários eventos, foi acordado que contássemos estes para o agrupamento de ***session_id*** e ***level_group*** (número de vezes que estes apareciam).

Consequentemente, ainda foram criadas colunas com dados novos, tal como "**ComprimentoTexto**", que conta o comprimento total do texto que aparece, por sessão, em cada grupo de níveis (é necessário referir que o grupo descobriu que existiam versões diferentes do jogo que possuíam frases e comentários diferentes para cada personagem, o que poderia influenciar o entendimento do jogo para os jogadores), daí ter incluído esta *feature*. Em baixo, é possível visualizar uma imagem daquilo que foi referido anteriormente.

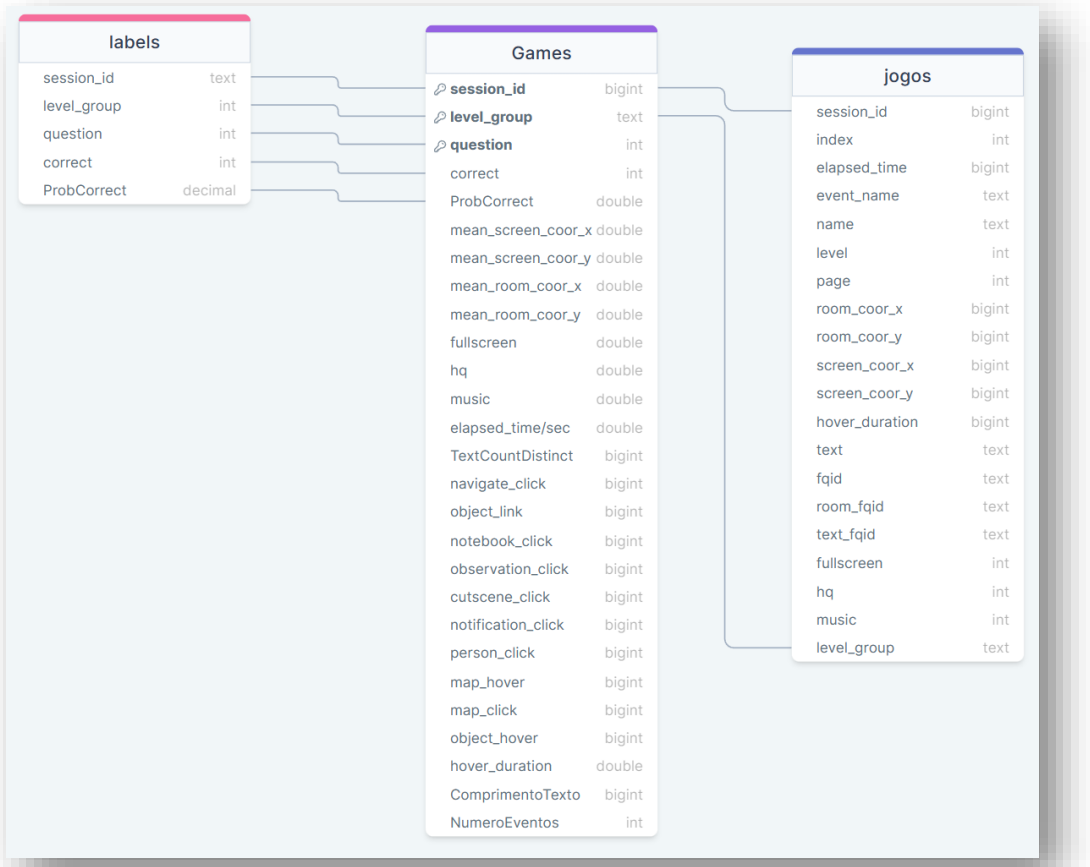



Figura 3 – Processo de criação da tabela ***GAMES***

Um ponto a melhorar da nossa preparação dos dados é a existência de dados repetidos. Por outras palavras, a única forma de realizar a divisão das perguntas (quando elas aconteceram) era através do "***level_group***", com os tais 3 grupos/ (*chapters* com perguntas). Devido a isso, para cada um destes grupos, as respetivas variáveis de cada jogador (Número de cliques, se jogou com música, etc…) aparecem várias vezes repetidos. Para tentar criar uma leve diferença entre perguntas, o grupo criou uma coluna que referia a percentagem de acertos em cada questão, já mencionada anteriormente.

# Dados não utilizados

Houve algumas variáveis que o grupo preferiu deixar de lado, pois, ou o grupo reparou que não teriam nenhum impacto no modelo, tal como "*room_fqid*" que tem o id de todas as salas do jogo, ou porque essas variáveis possuíam muitos nulos, tal como a coluna "*page*".

Somado a isto, é muito importante referir que a tabela *labels* não possuía as respostas de todas as ***session_id*** e, após uma breve discussão, optou-se por eliminar todos as ***session_id*** em que se desconheciam as respostas (com valores nulos).

# Visualização de dados

A componente de visualização/gráficos do trabalho foi realizada, especialmente, durante as fases de Leitura de Dados, onde estudámos algumas *features* que achámos interessante perceber com mais clareza, e de *Feature Engineering*, com as taxas de respostas corretas da tabela de *labels*. Com isto, queremos dizer que esta fase não seguiu uma ordem sequencial, mas que serviu como um ficheiro de apoio para o grupo no desenvolvimento do problema e na posterior criação de modelos.

Nesta fase também foram exploradas algumas features e, de seguida, foi criada a matriz de correlação, com variáveis numéricas, para justificar a decisão de escolha de *features* mais para a frente. Na tabela ***Games***, detetámos que não existia multicolinearidade com o alvo ***Correct***, pois todas tinham valores menores a 0.3 na correlação de Pearson.

# Outras formas que também poderiam resolver o problema

No processo da fase de *feature engineering*, nunca foi óbvia a proposta do que poderia ser feito, ou não, para a resolução dos problemas da base de dados descritos anteriormente. Ao longo desta fase, que consumiu algum tempo, tivemos algumas ideias do que poderia ser feito, como as apresentadas de seguida:

- Agrupar o ficheiro com todos os jogos e o ficheiro com as *labels*, mas definir as respostas às perguntas só nos momentos em que elas são respondidas (na sala do capitólio, com *notebook_click*). Decidimos não tomar este rumo porque estaríamos a rejeitar todos os outros tipos de eventos que são feitos na obtenção de pistas, mas também porque estes eventos não estavam tão nítidos na base de dados;
- Outra forma seria analisar todas as perguntas separadamente, criando modelos para cada pergunta. No entanto, só estaríamos a usar as mesmas perguntas para prever esse mesmo número da pergunta e a ignorar o facto dos *level_group*, e em que as mesmas eram respondidas no mesmo instante (algumas de cada vez).

# Ambientes de desenvolvimento/teste utilizados

Para a realização deste projeto o grupo utilizou, maioritariamente, *Pyspark* e *Python*, tendo passado maior parte do tempo no computador pessoal. No computador pessoal, o *Pyspark* foi utilizado através do *Jupyter Lab* no *Ubuntu* (tendo originado alguns problemas iniciais, possivelmente devido a atualizações). O grupo reparou que a tabela final, já trabalhada, limpa e preparada para se aplicar os modelos, não trazia uma quantidade excessiva de dados, mas, mesmo assim, o grupo apenas aplicou a

base de dados completa no *AWS*. Os modelos foram efetuados com uma amostra de 10% a nível local, só a caráter de teste.

## AWS

Na passagem para o *AWS*, o grupo deparou-se com algumas dificuldades, tendo sido, para além da fase de *feature engineering*, a fase que o grupo perdeu mais tempo a tentar resolver.

Num primeiro momento, o grupo deparou-se com um problema em relação aos ***imports*** e às bibliotecas (principalmente com o ***numpy***). Para o tentar resolver, foi necessário especificar qual a versão do ***numpy*** a utilizar, como ***numpy*==1.24.2** (versão mais recente)**.** Após atualizarmos a informação no ***bootstrap-script.sh*** diversas vezes, o problema persistiu e, então, decidimos apenas ler o script com os modelos.

De seguida, ocorreu um outro problema. O AWS não conseguia fazer nenhum ***count*()** nem aplicar os modelos com ***fit*()** e, após várias pesquisas, tentativas e erros, o grupo apercebeu-se que havia *features* com o ***type*()** errado. Para colmatar isto foi necessário criar um ***schema*** para a tabela ***Games***, de modo a solucionar este problema. Após isso, chegámos aos resultados descritos na secção seguinte.

# Resultados

De forma geral, o grupo ficou bastante surpreendido com os resultados dos modelos, tendo obtido previsões melhores face aquilo que esperava. O grupo tinha receio que a existência de vários dados repetidos pudesse ter um impacto muito negativo nestes, mas, pelos resultados, esse impacto negativo não pareceu ser tão crítico como acreditávamos ser.

Passando para os modelos em si, o grupo fez vários modelos com o objetivo de não só conhecer modelos novos, como também comparar os resultados entre estes. Apesar de termos mais do que um único modelo (4 ao todo), vemos que dois destes aproximam-se muito de modelos aleatórios. Um de Regressão Logística que caracterizava sempre os dados como positivos, no computador pessoal e *AWS*, talvez derivado do facto de que a probabilidade de a resposta estar correta ser muito superior à de estar incorreta (classes não balanceadas). O outro foi o ***Linear Support Vector Machine*** (*SVM*) que para a amostra tentava prever poucos negativos, tendo uma grande taxa de falha, tornando-se, claramente, um modelo quase aleatório na utilização de dados integrais no *AWS*. Acreditamos ser de o algoritmo não conseguir criar um *feature space* para os dados *raw*, visto o grande desequilíbrio das classes.

Ainda foi realizado um *Random Forest* que teve resultados muitos parecidos com o nosso melhor modelo, mas este apresentava uma menor qualidade em prever casos negativos e, por isso, referimos este modelo como sendo o nosso segundo melhor modelo.

Tendo já falado, de forma resumida, dos modelos com resultados não muito conseguidos, e do *Random Forest* que apresentou resultados bons, damos seguimento ao nosso melhor modelo, pois este é o que apresenta uma melhor robustez e uma maior qualidade em relação às suas métricas e resultados (*GBT Classifier*).

## *Gradient-boosted tree classifier*

De uma forma resumida, o ***Gradient-boosted tree classifier*** é um modelo de *machine learning* que utiliza uma técnica de *ensemble* (técnica de combinar vários modelos de *machine learning* para melhorar a precisão e robustez geral do modelo), para realizar a classificação de dados. Ele consiste em um conjunto de árvores de decisão simples, que são combinadas de forma sequencial para melhorar a

precisão da classificação. Cada árvore é treinada para corrigir os erros cometidos pela árvore anterior, resultando em um modelo robusto e preciso.

Dos modelos realizados, este foi o escolhido por apresentar resultados muito semelhantes, tanto na amostra, como no *dataset* completo, não falhando em aspetos fundamentais como os restantes.

# Amostra da base de dados, a nível local

De seguida, encontramos uma imagem da *confusion matrix* após aplicar o respetivo modelo numa amostra com 10% dos dados.

Tabela 2 – *Confusion Matrix* (Amostra)

| Confusion matrix | Actual positives | Actual negatives |
|---|---|---|
| Predicted positives | 5450 | 1702 |
| Predicted negatives | 420 | 694 |

Tendo finalmente a *confusion matrix*, extraímos que o modelo tem uma desproporção a classificar entre casos negativos e positivos, mas que na verdade tenta mesmo prever bastantes negativos, ao contrário de modelos anteriores. Veremos então o cálculo de algumas métricas de classificação, com o objetivo de realizar algumas análises.

```
Accuracy = 0.7440116138398258
Precison = 0.7627379619260918
Recall = 0.9282793867120954
Specificity = 0.2925709515859766
F1-score = 0.8374058706008912
```

Analisando os valores com atenção, podemos visualizar que este modelo tem uma precisão boa, atingindo um valor superior a 74% de acertos, tendo também uma boa capacidade preditora de positivos e uma não tão boa de negativos. O valor da área abaixo da curva de *ROC* (≈ 0.76) demonstra que este modelo é melhor do que uma previsão aleatória. Este algoritmo demorou cerca de 4 segundos a correr com uma amostra de 10%. Podemos estimar, de um modo grosseiro, que demoraria 10 vezes mais no AWS? Veremos!

# Base de dados completa

Nesta fase, iremos mostrar os resultados do mesmo algoritmo, mas ao aplicá-lo na base de dados inteira. É importante referir que nesta fase o algoritmo foi aplicado no *AWS*. De seguida, iremos ver a *confusion matrix* correspondente a este modelo com a base de dados:

Tabela 3 – *Confusion Matrix*

| Confusion matrix | Actual positives | Actual negatives |
|---|---|---|
| Predicted positives | 55507 | 17213 |
| Predicted negatives | 4071 | 7677 |

Ao observar a confusion matrix vemos que o modelo tem dificuldade a prever negativos e corretamente, pelo que acreditamos ter a ver com o desbalanceamento das classes de referência na base de dados. Passamos agora para o cálculo de algumas métricas de classificação com o objetivo de analisar o modelo mais atentamente.

```
Accuracy = 0.7480229199223375
Precison = 0.7632975797579759
Recall = 0.9316694081708013
Specificity = 0.3084371233427079
F1-score = 0.8391207728007983
```

Ao fazermos uma comparação entre os valores obtidos nas métricas, quando aplicado o modelo na amostra e na base de dados inteira, é possível concluir que os valores são muito parecidos. Dando uma maior ênfase à *specificity*, que teve um aumento, podemos assim concluir que houve uma melhoria mínima em relação á previsão de casos negativos, existindo um maior número de previsões corretas. De frisar também que o algoritmo demorou 13.74 segundos a rodar em *AWS*, mesmo levando em conta que a base de dados não era muito extensa (tempo muito menor ao expectável).

# Conclusão

Após a realização do projeto como um todo, conseguimos extrair bastantes conhecimentos e várias lições para o futuro. Dada uma base de dados com um tamanho consideravelmente grande, percebemos em como deveríamos agir para resolver o problema inicial: Como prever se as respostas dadas por jogadores estariam corretas ou incorretas. De vários pensamentos e análises, surgiu a ideia de preparação e modelação descrita neste relatório, apesar de reconhecermos que existem várias formas diferentes de pensar, tendo em conta que nenhuma delas poderia ser considerada totalmente certa ou totalmente errada.

Mesmo com a pequena dimensão dos nossos dados (devido à limpeza destes), conseguimos perceber tanto o intuito, como as vantagens, da utilização de um serviço *cloud* (*AWS*). Este dá-nos a possibilidade de processar os dados, de forma intensiva e eficiente, não sendo necessário alocar espaço na memória local dos computadores pessoais. Assim, concluímos que os resultados a que chegámos foram agradáveis, possuindo a consciência de que poderia existir alguma margem de melhoria.

Algumas destas melhorias já foram enunciadas no decorrer deste relatório, contudo, as perspetivas referidas levariam a termos de redesenhar o plano do modelo a aplicar.

A organização de ficheiros realizados ficou da seguinte forma:

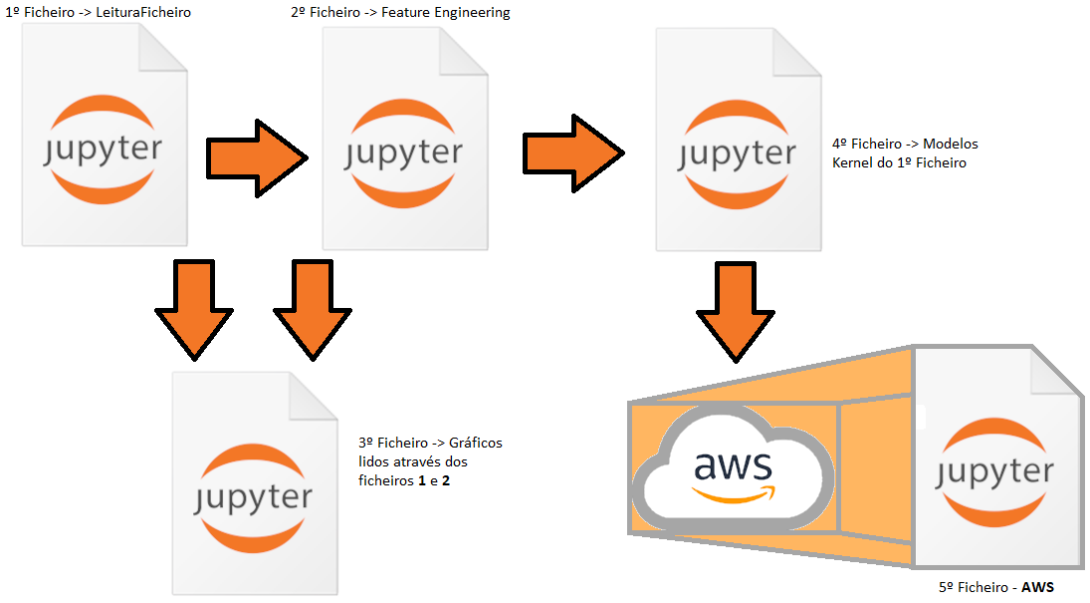



Figura 4 – A organização dos ficheiros