



**Universidade do Minho**

Mestrado Integrado em Engenharia Informática

## Projeto - Análise de Dados: Dataset Chess Game

PERFIL DE CIÊNCIA DE DADOS  
APRENDIZAGEM AUTOMÁTICA II

Diogo Dias - PG42825  
Nelson Costa - PG44587

*Docente*  
Miguel Rocha

Braga, Portugal  
7 de junho de 2021

# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>2</b>
<b>2</b>	<b>Exploração de Dados</b>	<b>2</b>
2.1	Dataset . . . . .	2
2.2	Missing Values . . . . .	3
2.3	Duplicated Values . . . . .	4
2.4	Exploração gráfica dos dados . . . . .	5
<b>3</b>	<b>Pré-Processamento</b>	<b>7</b>
<b>4</b>	<b>Modelos de Machine Learning</b>	<b>8</b>
4.1	Divisão do Dataset . . . . .	8
4.2	Construção de Modelos . . . . .	8
<b>5</b>	<b>Análise de Resultados</b>	<b>9</b>
5.1	K-Neighbors . . . . .	9
5.2	SVM . . . . .	9
5.3	Random Forest . . . . .	9
5.4	Adaptive Boosting . . . . .	9
5.5	MLP Model . . . . .	9
<b>6</b>	<b>Conclusão</b>	<b>10</b>

# 1 Introdução

No âmbito da unidade curricular de Aprendizagem Automática II foi proposta a realização de um trabalho prático, o qual tem como objetivos a análise de uma base de dados no ambiente python, em que as principais etapas são:

- Colheita/preparação de dados;
- Análise exploratória de dados;
- Pré processamento de dados;
- Construção de modelos;
- Otimização de hiperparâmetros;
- Predição através dos dados de teste;
- Avaliação do modelos.

Em Machine Learning, o computador aprende a realizar uma tarefa treinando um conjunto de dados. Em seguida, o computador realiza a mesma tarefa com os dados de teste. A aprendizagem supervisionada é de dois tipos: baseada em classificação e baseada em regressão. O problema apresentado neste projeto é supervisionado, uma vez que pretendemos efetuar uma medição da variável resposta (winner) que fornece a informação do vencedor (peças brancas, peças pretas, empate) num jogo de xadrez. Esta medição é realizada com base nas restantes variáveis (preditores). Portanto estamos a usar a aprendizagem supervisionada baseada em classificação.

A base de dados escolhida contém 20058 exemplos e 16 features. Através do pré processamento será verificada a existência de dados em falta, dados duplicados e posteriormente serão tratados esses mesmos dados. Para treinar a máquina, dividimos o conjunto de dados em duas partes (dados de treino e dados de teste). Esta divisão foi realizada assegurando que cada classe apresenta o mesmo número de exemplos. Assim a máquina irá treinar o modelo com o conjunto de dados de treino e posteriormente irá testar o conjunto de dados de teste. Segue-se a construção de modelos e a otimização de hiperparâmetros. Finalmente, é avaliado o modelo com a apresentação de métricas que permitam concluir se o modelo usado foi eficiente na tarefa de predição.

## 2 Exploração de Dados

### 2.1 Dataset

O *dataset Chess Game* foi retirado da plataforma Kaggle. O *dataset* é composto por 16 variáveis, entre as quais:

- *id*: identificação da partida;
- *rated*: identifica se a partida conta para o rating;
- *created\_at*: quando o jogo começou;
- *last\_move\_at*: quando o jogo acabou;

- *turns*: número de turnos que a partida teve;
- *victory\_status*: tipo de resultado da partida (empate, check mate, desistiu);
- *winner*: indica o vencedor, ou empate;
- *increment\_code*: número de segundos que são adicionados ao tempo depois de uma jogada;
- *white\_id*: identificação do jogador das peças brancas;
- *white\_rating*: classificação do jogador das peças brancas;
- *black\_id*: identificação do jogador das peças pretas;
- *black\_rating*: classificação do jogador das peças pretas;
- *moves*: todas as jogadas em notação de xadrex;
- *opening\_eco*: código que representa uma opening (movimento de abertura) em xadrex (lista de todas as openings);
- *opening\_name*: nome do movimento de abertura;
- *opening\_ply*: número de jogadas na fase de opening.

## 2.2 Missing Values

De modo, a detetar os valores em falta, foi elaborado um algoritmo de forma a percorrer as linhas de cada coluna e a validar se existem missing values. Esta validação apresenta para cada atributo a frequência relativa percentual de dados completos. Pela análise da figura, observamos que todos os campos estão populados, ou seja não existem missing values no *dataset*.

```

Verifica o quanto populados as colunas estão:
id está populado: 100.0 %
rated está populado: 100.0 %
created_at está populado: 100.0 %
last_move_at está populado: 100.0 %
turns está populado: 100.0 %
victory_status está populado: 100.0 %
winner está populado: 100.0 %
increment_code está populado: 100.0 %
white_id está populado: 100.0 %
white_rating está populado: 100.0 %
black_id está populado: 100.0 %
black_rating está populado: 100.0 %
moves está populado: 100.0 %
opening_eco está populado: 100.0 %
opening_name está populado: 100.0 %
opening_ply está populado: 100.0 %

```

Figura 1: Verificação de Missing Values

## 2.3 Duplicated Values

Através da verificação do número de dados duplicados observamos que existem 945 jogos com o id duplicado no *dataset*.

```
dupe = cd.duplicated(subset=['id'])
sum(dupe)

945
```

Figura 2: Verificação de jogos com id duplicado

De seguida pretende-se eliminar esses mesmos jogos:

```
cd = cd.drop_duplicates(subset=['id'])
```

Figura 3: Remoção de jogos duplicados através do id

Foi criado um algoritmo para validar a unicidade das features. Este algoritmo produz como resultado a percentagem em que os dados são únicos, isto é não produzem dados repetidos.

```
for column in cd.columns:
    if cd[column].is_unique:
        print(column, "is unique: 100%")
    else:
        comp1 = len(cd[column]) # número de linhas
        comp2 = len(cd[column].drop_duplicates(keep=False))
        print(column, "is unique: ", comp2/comp1 * 100, "%")

id is unique: 100%
rated is unique: 0.0 %
created_at is unique: 56.908910165855694 %
last_move_at is unique: 57.21760058598859 %
turns is unique: 0.1046408203840318 %
victory_status is unique: 0.0 %
winner is unique: 0.0 %
increment_code is unique: 0.5075079788625543 %
white_id is unique: 39.41296499764558 %
white_rating is unique: 0.7900381938994401 %
black_id is unique: 39.04149008528227 %
black_rating is unique: 0.8475906451106577 %
moves is unique: 98.72861403233402 %
opening_eco is unique: 0.26683409197928115 %
opening_name is unique: 2.0666562025846282 %
opening_ply is unique: 0.010464082038403181 %
```

Figura 4: Unicidade das features

Podemos retirar as seguintes informações da figura:

- 39% dos jogadores só jogaram uma vez de peças brancas ou só uma vez de peças pretas.

- 98% das jogadas ('moves') foram únicas, ou seja, em 2% dos jogos há partidas que tiveram moves iguais a outros.
- 2% de Movimentos de Abertura ('opening\_name') são únicos, só foram jogados uma vez no historial de 20k jogos.

## 2.4 Exploração gráfica dos dados

Com recurso ao gráfico circular averiguamos que a percentagem de jogos competitivos representa uma fatia de 80,92% de todas as partidas. No histograma, verifica-se que a frequência de turnos concentra-se em torno das 50 jogadas.

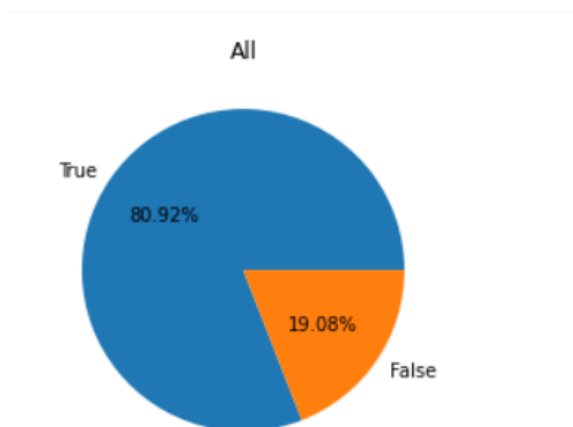


Figura 5: Gráfico circular - Jogos Competitivo (True) vs Jogos Casuais (False)

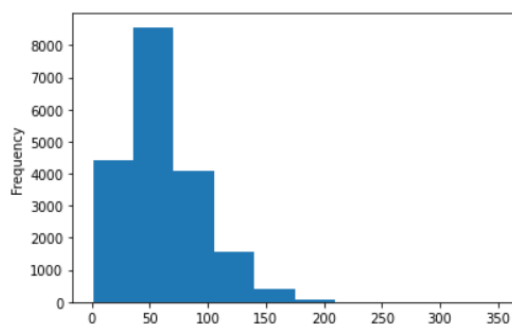


Figura 6: Frequência de turnos

O gráfico da figura 7 representa a percentagem de cada tipo de resultado de uma partida, e o gráfico da figura 8 representa a percentagem de vitórias de cada peça ou se é empate. No gráfico da figura 7 conseguimos observar que em 56% dos jogos eles acabam com um jogador a desistir, apenas 31% é que acabam em xeque mate. No gráfico da figura 8 vemos que no nosso dataset 50% das

partidas acabam com as peças brancas a ganharem, apenas 45% são vitórias para as peças pretas. Esta situação pode estar relacionada com o facto de as peças brancas terem o primeiro movimento numa partida.

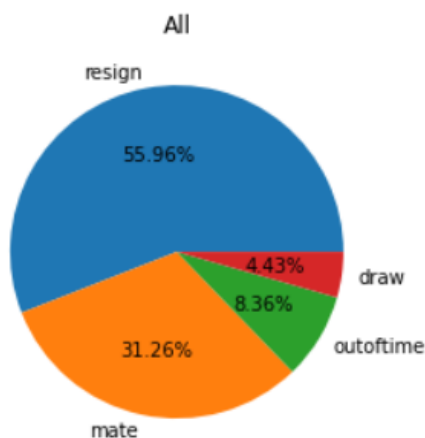


Figura 7: Percentagem de cada tipo de resultado de uma partida

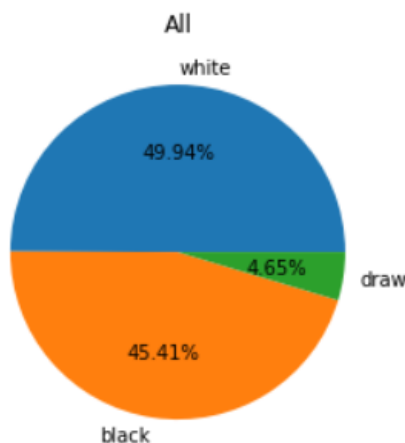


Figura 8: Percentagem de vitórias de cada peça (ou empate)

Aplicando a matriz de correlação aos dados conseguimos analisar que: *Created\_at* e *Last\_move\_at*, features que é suposto representarem a data de início e fim do jogo, apresentam correlação muito forte. *White\_rating* e *Black\_rating* têm correlação forte, o que faz sentido, o site prioriza partidas em que os jogadores têm rating semelhante. *Opening\_ply* expõe uma correlação fraca com o rating dos jogadores.

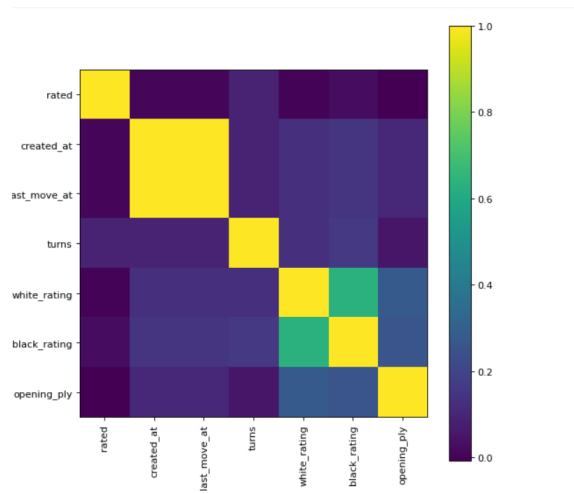


Figura 9: Matriz de Correlação

### 3 Pré-Processamento

De seguida torna-se fundamental remover as variáveis que não são interessantes para o modelo. O objetivo do trabalho é prever quem é o vencedor de uma partida de xadrex, ou então se a partida resultou em empate, através dos movimentos que os jogadores realizam ao longo do jogo.

As seguintes features foram removidas com o objetivo de desenvolver o melhor modelo de machine learning:

- *id*: Não tem nenhum impacto na partida;
- *rated*: Em princípio não tem nenhum impacto na partida;
- *created\_at last\_move\_at*: Não tem nenhum impacto na partida
- *turns*: Produz um modelo que é contra produtivo aos nossos objetivos, pois pelo número de turnos torna-se bastante fácil prever o resultado, visto que numa partida começa sempre o jogador das peças brancas. Por exemplo, numa partida que acabou em xeque-mate, se o número de turnos for ímpar, 13 por exemplo, foi o jogador das peças brancas o último a jogar, portanto presume-se que haja uma maior probabilidade de ter sido ele a ganhar. Se for um número par de turnos foi o jogador das peças pretas o último a jogar e é ele o presumido winner;
- *victory\_status and incrementcode*: Não nos ajuda a descobrir quem é o vencedor;
- *white\_id black\_id*: Não nos interessa saber quem é o jogador;
- *moves*: Pelo mesmo raciocínio que levou a retirar a feature 'turns', também não vai ajudar no modelo.



Copia-se a coluna 'winner' para uma variável  $y$  e de seguida realiza-se uma operação de label encoding, que passa os valores de um tipo de variável categórica para um valor numérico. Assim consegue-se usar essa variável como target do modelo de Machine Learning.

```
le = LabelEncoder()
y = le.fit_transform(y)
y = pd.DataFrame({'winner': y})
```

Figura 10: Operação de label encoding

As features *opening<sub>eco</sub>* e *opening<sub>name</sub>* são convertidas para variáveis fictícias (dummy variable) que indicam a presença ou ausência de algum efeito categórico que possa ter impacto na feature 'winner'.

## 4 Modelos de Machine Learning

Exposto nos capítulos abaixo temos o desenvolvimento dos modelos de aprendizagem supervisionada para a predição do vencedor de uma partida de Xadrez.

### 4.1 Divisão do Dataset

Apos normalizarmos os dados fazemos uma divisão dos dados, 70% para treino e 30% para teste e de seguida averiguamos sobre a frequência das classes da *target* feature. Reparamos que existe uma desproporcionalidade dos valores, mais especificamente o valor de empate, representando apenas 4% dos dados. Assim optamos por utilizar o método *SMOTE*, método de oversampling, para popular os dados e chegar a um equilíbrio. A razão disto foi de querermos evitar dar overfitting do modelo para a predição de vitória das peças brancas ou pretas enquanto que paralelamente, têm má precisão sobre resultados de empate.

### 4.2 Construção de Modelos

Os modelos de Machine Learning que vamos usar são:

- K-Neighbors
- Support Vector Machine
- Random Forest Classifier
- Adaptive Boosting

Numa primeira fase corremos os modelos com os default parameters de cada modelo, usando 5 – *Fold CrossValidation* para a previsão de resultados.

Posteriormente, usamos a biblioteca GridSearchCV para percorrer através de um fixed value de parameters que definimos, em que o resultado enquadra as configurações ótimas para o *estimator* (modelo de ML) a ser usado. Assim, com esta Otimização de Hiper-Parâmetros conseguimos melhorar o valor de precisão dos nossos modelos.

## 5 Análise de Resultados

### 5.1 K-Neighbors

Default	ANOVA	Otimizado
0.5983	0.4777	0.5466

### 5.2 SVM

Default	ANOVA	Otimizado
0.5783	0.4851	0.6054

### 5.3 Random Forest

Default	ANOVA	Otimizado
0.5814	0.5827	0.5905

### 5.4 Adaptive Boosting

Default	ANOVA	Otimizado
0.6198	0.6088	0.6321

### 5.5 MLP Model

Accuracy	Loss	Epochs	Batch.Size
0.53	0.5925	25	150

## 6 Conclusão

Com a realização deste trabalho conseguimos expandir os nossos conhecimentos na área de machine learning, mais especificamente conseguimos aplicar modelos para aprendizagem supervisionada, e conseguimos desenvolver a nossa capacidade de construir os ditos modelos, melhorar a qualidade deles através do uso de técnicas de oversampling, de otimização de hiper-parâmetros, mas também melhorar as nossas habilidades em termos de exploração de dados e pré-processamento destes. Apesar de os resultados de predição dos modelos serem apenas razoáveis, não é possível melhorar muito os modelos, possivelmente a única maneira seria com a inclusão de mais dados. Uma possível maneira de termos realizado o trabalho seria de expandir sobre o pré-processamento da feature *moves*, em que se separássemos por primeiro move, segundo move, etc, seria possível fazer uso de redes neurais para resolver o problema. Por exemplo, usando *LSTM's* ou *1D CNN's*, versão *Univariate* (só usando a feature *moves*, em que na sequência o primeiro elemento seria o primeiro move), e transformado os dados numa série temporal, seria uma possibilidade de usarmos modelos de deep learning no nosso dataset.