UNIVERSIDADE ESTADUAL DE MARINGÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE INFORMÁTICA

SEGUNDO TRABALHO PRÁTICO

APRENDIZADO DE MÁQUINA E
MODELAGEM DE CONHECIMENTO INCERTO

Acadêmicos:	RA:
Allan Diamante de Souza	105423
Felipe Diniz Tomás	110752

Prof. Dr. Wagner Igarashi

Maringá, 05 de maio de 2022

SUMÁRIO

Fundamentação teórica

Este capítulo conterá as seções explicativas do jogo de xadrez, que incluem as regras básicas do jogo, sua origem, além das técnicas possíveis de serem aplicadas nas partidas, como as jogadas de abertura que são comuns em partidas profissionais de xadrez.

O jogo de xadrez

O xadrez é um jogo de tabuleiro estratégico jogado entre dois jogadores, tendo originado-se a quase 1.500 anos atrás, com seu predecessor mais antigo conhecido como *chaturanga*, jogado na Índia. Especula-se que da Índia se espalhou para a Pérsia e após a invasão árabe e a conquista da Pérsia, o xadrez foi adotado pelo mundo muçulmano e posteriormente se espalhou para o sul da Europa (SHENK 2007). Como sabemos o jogo evolui muito nos últimos anos, se tornando cada vez mais popular, e adotando mudanças e adição de regras.

O jogo em si, se desenvolveu muito durante o século XIX. Foi neste período em que ocorreram partidas e competições oficiais de xadrez, onde Wilhelm Steinitz tornou-se o primeiro campeão mundial oficial de xadrez em 1886. Atualmente com a evolução da teoria do xadrez, que consiste em conhecimento de abertura, táticas, análise posicional, estratégia e técnica de final de jogo, grandes campeões surgiram e tornaram o xadrez uma grande disputa de técnica e pensamento lógico.

Uma figura bastante emblemática e presente no cenário mundial, foi Garry Kasparov o jogador mais novo a se tornar campeão mundial de xadrez em 1985 aos 22 anos, mantendo o título mundial de melhor jogador oficial da Federação Internacional de Xadrez até 1993. Protagonizou também uma das primeiras partidas jogadas entre um computador e um humano em 1997, sendo derrotado pelo então computador mais potente *Deep Blue* da gigante americana IBM.

Desenvolvimentos posteriores no século 21 tornaram o uso da análise computacional muito superior à capacidade de qualquer jogador humano acessível ao público. Os jogos online, que apareceram pela primeira vez em meados da década de 1990, também se tornaram populares no século 21 e com eles surgiram

plataformas para se jogar xadrez com diferentes jogadores ao redor do mundo pela internet.

Regras do xadrez

O xadrez é jogado em um tabuleiro com 64 casas dispostas em uma grade de oito por oito. No início, cada jogador (um controlando as peças brancas, o outro controlando as peças pretas) controla dezesseis peças: um rei, uma rainha, duas torres, dois bispos, dois cavalos e oito peões. O objetivo do jogo é dar xeque-mate ao rei adversário, onde o rei está sob ataque imediato (em “xeque”) e não há como escapar. Existem também várias maneiras de um jogo terminar empatado. Além dos movimentos básicos das peças, as regras também regem o equipamento utilizado, controle de tempo, conduta e ética dos jogadores. Procedimentos para resolver irregularidades que podem ocorrer durante um jogo também são fornecidos.

No início do jogo, as peças são dispostas conforme mostrado na figura 1, para cada lado um rei, uma rainha, duas torres, dois bispos, dois cavalos e oito peões. As peças são colocadas nas extremidades do tabuleiro preenchendo as duas primeiras fileiras de cada jogador.

Figura 1: Tabuleiro de xadrez inicial

Fonte: Chess.com

O jogador que controla as peças brancas é chamado de "Branco"; o jogador que controla as peças pretas é chamado de "Preto". As brancas se movem primeiro, depois os jogadores alternam os movimentos. Fazer um movimento é necessário,

mesmo quando ter que se mover é prejudicial. O jogo continua até que um rei dê xeque-mate, um jogador renuncie ou um empate seja declarado, conforme explicado abaixo. Além disso, se o jogo estiver sendo jogado sob controle de tempo, um jogador que exceder o limite de tempo perderá o jogo, a menos que não possa receber xeque-mate.

As regras oficiais de xadrez não incluem um procedimento para determinar quem joga com as brancas. Em vez disso, essa decisão é deixada em aberto para regras específicas do torneio ou, no caso de jogo não competitivo, acordo mútuo, caso em que algum tipo de escolha aleatória é frequentemente empregada.

Cada tipo de peça de xadrez tem seu próprio método de movimento. Uma peça se move para uma casa vazia, exceto ao capturar uma peça do oponente. O rei, por exemplo, se move exatamente uma casa na horizontal, vertical ou diagonal. Também existem movimentos especiais, que trazem nova dinâmica ao jogo, como é o caso do chamado roque. As combinações de movimentos em determinada fase do jogo são conhecidas e fazem parte da teoria do xadrez, já que garante uma estratégia de jogo em determinado cenário e garante resultados já conhecidos.

Técnicas de jogadas

Diversas técnicas de jogadas podem ser aplicadas em determinada fase de uma partida de xadrez, uma delas bastante conhecida é a sequência de movimentos chamada de abertura, que ocorre no início da partida. Segundo estudos, os primeiros lances de uma partida de xadrez podem ser decisivos quanto ao resultado da partida. Com estes lances é estabelecido o começo dos planos e luta de cada jogador para colocar suas peças no tabuleiro.

Logo, a abertura no xadrez são sequências de movimentos iniciais que já foram muito testadas em tabuleiros reais e em partidas virtuais, com as quais o jogador pode ter certa vantagem sobre um adversário. Ao memorizar algumas das principais aberturas e suas variações é possível jogar os primeiros movimentos de uma maneira quase automática e assim focar-se nos pontos mais importantes do jogo.

Normalmente utiliza-se o termo "aberturas de xadrez" apenas para os movimentos das peças Brancas, enquanto para as peças Pretas chama-se a

sequência de movimentos de "defesa". No entanto, na prática, as defesas são como aberturas de xadrez para as peças pretas e algumas aberturas até podem ser usadas como defesas.

A Abertura Italiana é uma das mais recomendadas a iniciantes, começa com 1.e4 e5 2.Nf3 Nc6 3.Bc4, como pode ser visto na figura 2, que apresenta o resultado final dessa configuração.

Figura 2: Abertura Italiana

Fonte: Chess.com

Esta abertura preza por desenvolvimento rápido e controle do centro com o Bispo de casas brancas, em apenas três movimentos o jogador com as peças brancas consegue colocar uma peça forte no centro, difícil de ser atacada, e está pronto para efetuar o Roque pequeno no próximo movimento.

Outra famosa abertura, é a chamada Ruy-Lopez, começa com 1.e4 e5 2.Cf3 Cc6 3.Bb5, como é possível observar na figura 3, que apresenta o resultado final dessa configuração.

Figura 2: Abertura Ruy-Lopez

Fonte: Chess.com

Apesar de muito parecida com o abertura Italiana, com uma mera diferença da casa na qual o Bispo repousa, essa abertura leva para posições muito diferentes e costuma fazer com que jogadores amadores se sintam pressionados logo cedo. O Bispo ameaça imediatamente o cavalo, e essa tensão pode ser deixada de lado até mais tarde na partida. Em outras defesas, no entanto, as peças pretas podem ameaçar o Bispo com o ataque do Peão de A7. Neste caso o Bispo pode recuar e as peças brancas perdem um pouco de Tempo, ou podem capturar o Cavalo.

Existem diversas outras aberturas e cada uma configura um estado de partida diferente, com diferentes combinações de movimentos, inclusive de defesa para peças pretas, protegendo-se da iniciativa das brancas nesse cenário.

Metodologia

Neste capítulo será definido a metodologia, com seções explicando o *dataset* utilizado e seu tratamento, as técnicas de modelo escolhidas, além das dependências de implementação e a escolha de outro modelo implementado por alguém para fins de comparação.

Dataset adotado

Para a pesquisa do *dataset* escolhido, foi utilizado o site *Kaggle*[1] que possui uma base gigantesca com diversos banco de dados e código de implementações empregando os mesmos. O dataset selecionado foi *Chess Game Dataset (Lichess)*[2], produzido pelo autor MITCHELL J, que contém um conjunto de mais de 20.000 jogos coletados de uma seleção de usuários no site *Lichess*[3].

Lichess é um website focado em jogos de xadrez. Qualquer um pode jogar anonimamente, embora os jogadores possam registrar uma conta no site para jogar jogos classificados. Todos seus serviços estão disponíveis gratuitamente, pois o site é financiado por meio de doações (DUPLESSIS, 2014). Para jogadores registrados, *Lichess* emprega um sistema de classificação *Glicko-2* e concede a capacidade de competir em torneios, postar nos fóruns e solicitar uma análise completa ao servidor para qualquer jogo finalizado.

O dataset possui originalmente 20056 linhas e um total de 16 colunas definidas da seguinte forma:

- *Game ID* (Identificador da partida)*;*
- *Rated (T/F)* (Se é uma partida ranqueada)*;*
- *Start Time* (Horário de início);
- *End Time (*Horário de término*);*
- *Number of Turns* (Número de turnos);
- *Victory Status* (Status da finalização da partida);
- *Winner* (Campeão)*;*
- *Time Increment* (tempo de incremento de cada turno)*;*
- *White Player ID (*Identificador do player com as peças brancas*);*
- *White Player Rating* (Ranque do player com as peças brancas);
- *Black Player ID* (Identificador do player com as peças pretas);
- *Black Player Rating* (Ranque do jogador com peças pretas);
- *All Moves in Standard Chess Notation* (Todos os movimentos da partida); ,
- *Opening Eco* (Código identificador de tipo de abertura);
- *Opening Name* (Nome da abertura)*;*

[1] https://www.kaggle.com/
[2] https://www.kaggle.com/datasets/datasnaek/chess
[3] https://lichess.org/

- *Opening Ply (*Quantidade de movimentos feitos relacionados a abertura*).*

Tratamento do *dataset*

Para realizar o pré-processamento do *dataset*, foi definido quais das colunas seriam utilizadas, nesse caso, foi observado que algumas dessas colunas poderiam não ser relevantes ao resultado. Por exemplo, *Black Player ID* e *White Player ID* não contribuem para o modelo, assim como *Game ID, Start Time e End Time*. A coluna *moves* também não foi considerada já que a normalização de tais dados seria muito complexo já que apresenta toda a configuração de movimentos em notação algébrica de xadrez de cada turno. Sendo assim, foi considerado 10 colunas, sendo renomeadas para fim de compressão:

- *victory_status* = finalizacao_partida
- *white_rating* = ranque_brancas
- *black_rating = ranque_pretas*
- *opening_eco = abertura_codigo*
- *opening_name = nome_abertura*
- *opening_ply* = movimentos_abertura
- *increment_code* = incremento_turno
- *turns* = turnos
- *rated* = ranqueada
- *winner* = ganhador

Foi criado também a coluna “diferenca_pontos” que nada mais é que a diferença de ranque entre os dois jogadores participantes da partida. Além disso observou que 9 colunas do *dataset* possuem os valores no formato *object*, sendo assim foi necessário utilizar a técnica de *Label Encoder* fornecido por uma biblioteca, que basicamente atribui valores inteiros como *labels* para *strings* de forma única para cada uma.

Também foi necessário realizar uma normalização de classes na coluna “ganhador", que é a coluna alvo. Nesse caso, os valores possíveis para esse atributo é 0 (jogador com peças pretas ganhou), 1 (empate), 2 (jogador com peças brancas ganhou), assim aplicando a função SMOTE de uma biblioteca externa, foi possível aplicar uma distribuição igualitária para essa classe. Por fim, também foi empregado

o *MinMaxScaler* de uma biblioteca externa, com intuito de normalizar os valores da coluna que serviram como parâmetros dos algoritmos de modelo de aprendizado de máquina.

Modelos de aprendizado de máquina

Nesta etapa foram definidos dois modelos de aprendizado de máquina KNN(*K — Nearest Neighbors*) e *Random-forest*, esta seção explica um pouco de cada um deles.

K-Nearest Neighbors

O algoritmo KNN(*K-Nearest Neighbor*s) é um dos muitos algoritmos (de aprendizagem supervisionada) usado para resolver problemas de classificação e regressão. Sua implementação não é complexa, porém sua velocidade de execução tende a diminuir de acordo com o tamanho da base de dados utilizada.

A KNN faz previsões usando o conjunto de dados de treinamento diretamente. As previsões são feitas para uma nova instância (x) pesquisando todo o conjunto de treinamento para as K instâncias mais semelhantes (os vizinhos) e resumindo a variável de saída para essas instâncias de K. Para a regressão, essa pode ser a variável de saída média; na classificação, esse pode ser o valor de classe do modo (ou mais comum). Os parâmetros dessa implementação incluem:

- ***n_neighbors***: Número de vizinhos usados por padrão.
- ***leaf_size***: Pode afetar a velocidade de construção e busca, tanto como a memória necessária para armazenar a árvore.
- ***weights***: Função de peso utilizada na predição, os valores possíveis são:
    - *uniform*: Todos os pontos em cada vizinhança tem o mesmo peso;
    - *distance*: O peso e o inverso de sua distância.
- **P**: Parâmetro de potência para a métrica de Minkowski.
    - p = 1 é equivalente ao uso de manhattan distance
    - p = 2 é equivalente à distância euclidiana.

Random Forest

Em português, *Random Forest* significa floresta aleatória. Este nome explica muito bem o funcionamento do algoritmo, que irá criar muitas árvores de decisão, de maneira aleatória, formando o que podemos enxergar como uma floresta, onde cada árvore será utilizada na escolha do resultado final.

É um método para classificação, regressão e outras tarefas que opera construindo uma infinidade de árvores de decisão em tempo de treinamento. Para tarefas de classificação, a saída da floresta aleatória é a classe selecionada pela maioria das árvores. Para tarefas de regressão, a previsão média ou média das árvores individuais é retornada. As florestas de decisão aleatória corrigem o hábito das árvores de decisão de realizar "*overfitting*" com seu conjunto de treinamento. Os parâmetros dessa implementação incluem:

- ***n_estimators***: O número de árvores na floresta.
- ***max_depth***: A profundidade máxima da árvore. Se for "*None*" os nós são expandidos até que todas as folhas sejam puras ou contenham menos que "*min_samples_split*" amostras.
- ***max_features***: O número de características a se considerar quando se busca o melhor *split*. Pode ser um valor inteiro, de ponto flutuante, logarítmico.
- ***min_samples_leaf***: O valor mínimo de amostras necessárias em um nó folha. Um ponto de *split* em qualquer profundidade só será considerado caso sobre pelo menos "*min_samples_leaf*" amostras de treinamento em cada um dos ramos esquerdo e direito.
- ***bootstrap***: Caso "True", amostras são usadas na construção de árvores. Se "*False*" o *dataset* inteiro é utilizado na construção de cada árvore.

Dependências de implementação

As dependências do projeto incluem as bibliotecas:

- *Pandas* [4] (Criação do Dataframe);

[4] https://pandas.pydata.org/

- *Matplotlib*[5] (Plot de Gráficos);
- *Seaborn* [6] (Visualizações de Dados);
- *Numpy* [7] (Operações matemáticas);
- *Imblearn* [8] (Classificação e Balanceamento);
- *Scikit* [9] (Aprendizado de máquina);

Implementação externa para comparação

Como exigido, foi necessário buscar uma implementação que utilizasse o mesmo *dataset* e também realizasse uma avaliação de implementação de modelos de aprendizado de máquina para fins de comparação com o nosso modelo. Neste caso, o próprio site *Kaggle* possui uma aba no *dataset* que inclui várias implementações utilizando o mesmo.

Logo, através de um filtro de busca, encontramos poucos resultados que implementaram especificamente o que queríamos, porém uma das implementações "*Lichess: predicting a winner*"[10] Possui métricas de avaliação para dois modelos, um de regressão logística e árvore de decisão. E portanto, mesmo sendo de modelo diferentes, serve como comparação.

Hyper-parametrização

A Hyper-parametrização é uma técnica de testagem de vários modelos, até encontrar o qual com a melhor resultado, alterando apenas os parâmetros. A biblioteca *Scikit* possui a implementação das duas técnicas, nesse caso foram utilizadas as duas técnicas para os diferentes modelos. O *K-Nearest Neighbor*s utilizou grid search enquanto *Random Forest* utilizou *random search*.

No *Grid Search*, define um espaço de pesquisa como uma grade de valores de hiperparâmetros e avalia cada posição na grade. Já o *Random Search* define um espaço de busca como um domínio limitado de valores de hiperparâmetros e pontos

[5] https://matplotlib.org/
[6] https://seaborn.pydata.org/
[7] https://numpy.org/
[8] https://imbalanced-learn.org/
[9] https://scikit-learn.org/
[10] https://www.kaggle.com/code/nelver/lichess-predicting-a-winner

de amostra aleatórios nesse domínio. O *grid search* é ótimo para combinações de verificação pontual que são conhecidas por terem um bom desempenho em geral. O *random search* é ótimo para descobrir e obter combinações de hiperparâmetros que você não teria adivinhado intuitivamente, embora muitas vezes exige mais tempo para executá-las.

Avaliação do Modelo

Neste capítulo é mostrado a avaliação dos modelos através da métrica *F1-Score,* que é uma medida da precisão, que é calculada a partir das métricas de precisão e *recall*, onde a precisão é o número de resultados positivos verdadeiros dividido pelo número de todos os resultados positivos, incluindo aqueles não identificados corretamente, e o recall é o número de resultados positivos verdadeiros dividido pelo número de todas as amostras que deveriam ter sido identificadas como positivas.

Além disso, também é realizada a comparação do modelo que foi implementado por outro autor pego no site com *Kaggle*.

K-Nearest Neighbors

O melhor parâmetro encontrado para o modelo KNN foi: *n_neighbors*: [1, 5, 10, 25, 50, 750]; *weights*: ['distance','uniform']; *leaf_size*: [1, 5, 10, 25, 50, 750]; p: [1,2]; A figura 3 mostra os resultados obtidos da métrica *F1-score* para o modelo KNN.

Figura 3: Resultado *F1-score* para o KNN.

K-Nearest Neighbors

	precision	recall	f1-score	support
0	0.59	0.68	0.63	2719
1	0.76	0.95	0.85	300
2	0.67	0.56	0.61	2999
accuracy			0.64	6018
macro avg	0.68	0.73	0.70	6018
weighted avg	0.64	0.64	0.63	6018

Fonte: o Autor.

Random Forest

O melhor parâmetro encontrado para o modelo KNN foi: *n_estimators*: [4, 200]; *max_features*: truncnorm(a=0, b=1, loc=0.25, scale=0.1); *max_depth*: [2 ,5 ,10]; *min_samples_split*: [10, 50, 100]; A figura 4 mostra os resultados obtidos da métrica *F1-score* para o modelo KNN.

Figura 4: Resultado *F1-score* para o Random Forest.

Random Forest

	precision	recall	f1-score	support
0	0.61	0.70	0.65	2719
1	0.94	0.96	0.95	300
2	0.69	0.59	0.64	2999
accuracy			0.66	6018
macro avg	0.75	0.75	0.75	6018
weighted avg	0.67	0.66	0.66	6018

Fonte: o Autor

Comparação entre as implementações

A figura 5 e 6 mostram o resultado do autor encontrado na internet, para os modelos de regressão logística e árvore de decisão.

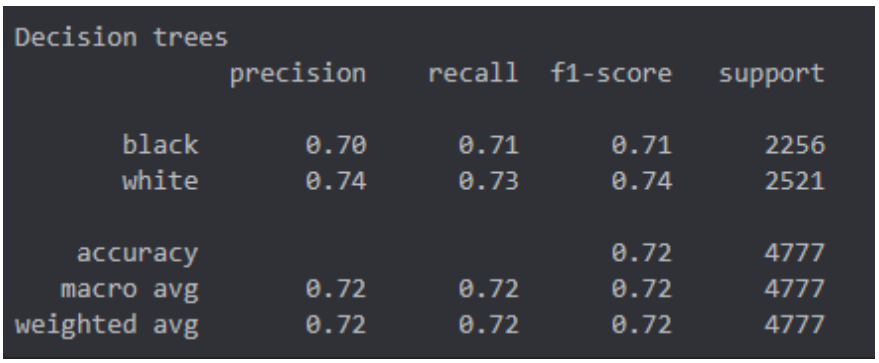
Figura 5: Resultado *F1-score* para a Regressão logística.

Logistic regression

	precision	recall	f1-score	support
black	0.63	0.61	0.62	2256
white	0.66	0.67	0.67	2521
accuracy			0.64	4777
macro avg	0.64	0.64	0.64	4777
weighted avg	0.64	0.64	0.64	4777

Fonte: Elver (2021)

Figura 6: Resultado *F1-score* para Árvore de decisão.



Decision trees

	precision	recall	f1-score	support
black	0.70	0.71	0.71	2256
white	0.74	0.73	0.74	2521
accuracy			0.72	4777
macro avg	0.72	0.72	0.72	4777
weighted avg	0.72	0.72	0.72	4777

Fonte: Elver (2021).

Observou-se que a implementação encontrada na internet, obteve 73% de acurácia no modelo de árvore de decisão e 64% de acurácia no modelo de regressão logística. É necessário pontuar que a implementação do autor transformou o problema com classe binária, ou seja, desconsiderou os empates, além disso ele não balanceou a base de dados, deixando vantagem para peças brancas, e ainda empregou outro método de normalização para aplicar os valores das colunas no modelo. Porém observa-se que a avaliação tem resultados de acurácia semelhantes, e a precisão não é tão alta.

Cenários de Uso

Neste capítulo são definidos os três cenários de uso para cada modelo, e os resultados obtidos conforme a tarefa de predição.

Cenário 1

Variáveis do cenário:

- ranqueada: *True*
- turnos: 13
- finalizacao_partida: *outoftime*
- incremento_turno: 15+2
- ranque_brancas: 1500
- ranque_pretas: 1191
- diferenca_pontos: 309

- nome_abertura: *Slav Defense: Exchange Variation*
- movimentos_abertura: 5
- abertura_codigo: D10
- ganhador: *white*

Resultado obtido no modelo KNN:

30% de Pretas vencerem. 0% de empate. 69,9% de Brancas vencerem.

Escolha do modelo: Brancas

Resultado obtido no modelo Random Forest:

39.7% de Pretas vencerem. 5.2% de empate.54.9% de Brancas vencerem.

Escolha do modelo: Brancas

Cenário 2

- ranqueada: *True*
- turnos: 16
- finalizacao_partida: *resign*
- incremento_turno: 5+10
- ranque_brancas: 1322
- ranque_pretas: 1261
- diferenca_pontos: 61
- nome_abertura: *Nimzowitsch Defense: Kennedy Variation*
- movimentos_abertura: 4
- abertura_codigo: B00
- ganhador: *black*

Resultado obtido no modelo KNN:

52,4% de Pretas vencerem. 0% de empate. 47,5% de Brancas vencerem.

Escolha do modelo: Pretas

Resultado obtido no modelo Random Forest:

42% de Pretas vencerem. 5% de empate. 52% de Brancas vencerem.

Escolha do modelo: Brancas

Cenário 3

- ranqueada: *True*
- turnos: 61
- finalizacao_partida: *mate*
- incremento_turno: 5+10
- ranque_brancas: 1496
- ranque_pretas: 1500
- diferenca_pontos: -4
- nome_abertura:*King's Pawn Game: Leonardis Variation*
- movimentos_abertura: 3
- abertura_codigo: C20
- ganhador: *white*

Resultado obtido no modelo KNN:

52.5% de Pretas vencerem. 0% de empate. 47.4% de Brancas vencerem.

Escolha do modelo: Pretas

Resultado obtido no modelo Random Forest:

52% de Pretas vencerem. 0% de empate. 47,8% de Brancas vencerem.

Escolha do modelo: Pretas

Referências

Shenk, David 2007. ***The Immortal Game: A History of Chess***. 2007, Knopf Doubleday. p. 99

DUPLESSIS, Thibault. **Why is lichess free**?: The founder and developer explains the philosophy behind lichess., 12 jul. 2014. Disponível em: https://lichess.org/blog/U4skkUQAAEAAhIGz/why-is-lichess-free. Acesso em: 3 maio 2022.

MITCHELL. Chess Game Dataset (Lichess), 2017. Disponível em: https://www.kaggle.com/datasets/datasnaek/chess Acesso em: 03 de maio de 2022.

NELVER. Lichess: predicting a winner, 2021. Disponível em: https://www.kaggle.com/code/nelver/lichess-predicting-a-winner Acesso em: 03 de maio de 2022