UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO DE DESPORTOS
DEPARTAMENTO DE EDUCAÇÃO FÍSICA

CRISTIANO ZARBATO MORAIS

DATA MINING NO ESPORTE: UM EXEMPLO DE APLICAÇÃO

Florianópolis
2022

Cristiano Zarbato Morais

DATA MINING NO ESPORTE:
UM EXEMPLO DE APLICAÇÃO

Trabalho de Conclusão do Curso de Graduação em Educação Física – Bacharelado do Centro de Desportos da Universidade Federal de Santa Catarina como requisito para a obtenção do Título de Bacharel em Educação Física.
Orientador: Prof. Dr. Humberto Moreira Carvalho
Coorientador: Prof. Ms. Ahlan Benezar Lima

Florianópolis
2022

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Morais, Cristiano Zarbato
Data mining no esporte : um exemplo de aplicação / Cristiano Zarbato Morais ; orientador, Humberto Moreira Carvalho, coorientador, Ahlan Benezar Lima, 2022.
52 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro de Desportos, Graduação em Educação Física, Florianópolis, 2022.

Inclui referências.

1. Educação Física. 2. Data Mining. 3. Python. 4. Big Data. 5. Esporte. I. Carvalho, Humberto Moreira. II. Lima, Ahlan Benezar. III. Universidade Federal de Santa Catarina. Graduação em Educação Física. IV. Título.

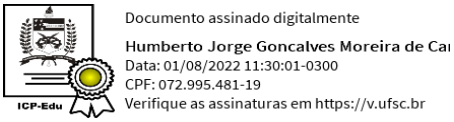
Cristiano Zarbato Morais

DATA MINING NO ESPORTE: UM EXEMPLO DE APLICAÇÃO

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de "Bacharel em Educação Física"e aprovado em sua forma final pelo Centro de Desportos da Universidade Federal de Santa Catarina, com a nota 10.

Florianópolis, 21 de julho de 2022.

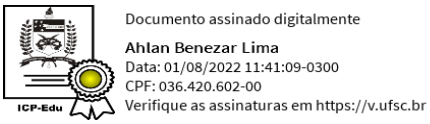
Banca Examinadora:



Prof. Humberto Moreira Carvalho, Dr.

Orientador

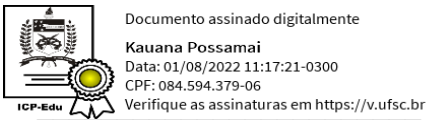
Universidade Federal de Santa Catarina



Prof. Ahlan Benezar Lima, Ms.

Corientador

Universidade Federal de Santa Catarina



Prof.ª Kauana Possamai

Universidade Federal de Santa Catarina

Eng. Gabriel Martins do Rosário

Hexagon Agriculture

Este trabalho é dedicado à minha família que sempre apoiou minhas decisões, e aos meus amigos com quem sempre posso contar.

AGRADECIMENTOS

Com toda a certeza, essa parte do trabalho é a mais difícil de escrever. É aqui que você para e pensa em toda a trajetória que percorreu até então, e relembra de todos aqueles que estavam lá por você. Aqueles que te ensinaram, te apoiaram, te fizeram rir, te convenceram a não desistir, ou que simplesmente passaram e deixaram alguma coisa que te fez evoluir. É aqui que você percebe que o mais importante não é a chegada, mas sim a caminhada.

De tantas pessoas que preciso agradecer por terem engrandecido minha caminhada até o objetivo, não poderia começar com outros senão minha família. Seu Duany e Dona Cristiani, meu pai e minha mãe, que me apoiaram de todas as formas, possíveis e impossíveis, e a Ju, minha irmã, que apesar de ter sua própria caminhada para percorrer, não abre mão de me ajudar sempre que eu preciso. A eles, que todos os dias me dão suporte e me suportam, o meu mais sincero obrigado. EU AMO VOCÊS! Ainda no âmbito familiar, menção honrosa pra Puca, que sempre esteve lá pedindo carinho e comida, e fazendo gracinhas para alegrar nosso dia.

Aos meus padrinho e madrinha, vô e vó, Seu Augusto e Dona Lenice, obrigado por sempre estarem dispostos a escutarem minhas histórias, por mais sem graça que elas sejam muitas vezes, e obrigado pelos ótimos almoços de domingo com muita risada. A Guta por fazer parte e aguentar a zoação, e ao Kim, por ser sempre solícito e ter me auxiliado e muito na execução desse trabalho.

A todos os meus amigos, mais próximos ou mais distantes, por terem em algum momento me auxiliado emocionalmente, sendo jogando algum jogo, praticando algum esporte ou mesmo tomando uma cerveja, vocês foram essenciais em todo esse processo, obrigado! A Duda, Dudu e Marlon, por terem sido minha companhia diária pelo Discord, durante toda a pandemia, seja pra bater um papo ou assistir um filme juntos, vocês me ajudaram a passar por uma fase bem difícil, obrigado!

A todos os colegas do Grupo de Pesquisa em Formação Desportiva e do Centro de Formação no Treino de Basquetebol: Ricardinho, Kauana, André, Jaque, Felipe e Caio, meu muito obrigado por contribuírem para minha formação. E por fim, obrigado ao Ahlan e ao professor Humberto, que me auxiliaram e me apoiaram na elaboração desse trabalho!

RESUMO

A tecnologia cada vez mais faz parte do nosso dia a dia e, tudo que fazemos geram informações, desde uma compra feita online, até um comentário em redes sociais. Dados têm um impacto direto na vida das pessoas, e mais do que nunca, diversas áreas e estruturas sociais estão utilizando os dados e informação disponível como ferramenta para auxiliar a tomada de decisão. No esporte não é diferente, já que a imensidão de dados gerados em uma competição esportiva ou durante os treinamentos, podem auxiliar treinadores nos diversos processos de decisão. Adicionalmente, a coleta, processamento e comunicação de dados no esporte têm permitido o envolvimento mais próximo de atletas, fãs e mídia também são beneficiados pelos conhecimentos providos pelos dados. Entretanto, o uso de ferramentas de programação que tornem a coleta, análise e visualização dos dados não estão ainda difundidas no âmbito esportivo. Observada a importância do assunto, esse trabalho tem como objetivo elaborar e apresentar um passo a passo de parte do processo de Data Mining no âmbito esportivo, utilizando programação.

Palavras-chave: Data mining. Python. Big Data. Esporte.

ABSTRACT

Technology is increansingly part of our lives and everything we do generates information, from online shopping to a comment on social media. Data has a direct impact on people's lives, and more than over, different areas and social structures are using data and information available as a tool to help in decision making. There's no difference in sports field, since the immensity of data generated in a sporting competition or during training sessions, can help coaches in the various decision processes. Additionally, the collection, processing and communication of data in sport has allowed the closer involvement of athletes, fans and the media who also benefit from knowledge provided by the data. However, the of programming tools that make collection, analysis and visualization of data are not yet widespread in the sports field. Observing the importance of the subject, this work aims to elaborate and present a step by step of part of the Data Mining process in the sports field, using programming.

Keywords: Data mining. Python. Big data. Sport.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Todos os dias, uma quantidade enorme de dados é gerada, desde *e-mails*, *sites* de redes sociais com os vídeos no *Youtube* ou publicações no *Facebook*, até dados de geolocalização e sequenciamento genético (DIETRICH; HELLER; YANG, 2015; PRAVEENA; BHARATHI, 2017). No ritmo atual, a previsão é de que até 2025, mais de 150 *zettabytes* (10^{21} *bytes* ou 150 trilhões de *gigabytes*) será o volume de dados existente que precisará ser analisado em todo o mundo (KULKARNI, 2019).

Esse grande amontoado de dados, grande o suficiente para tornar-se complexo a ponto de não poder ser processado por métodos convencionais, é denominado *Big Data*, e existe na literatura um consenso sobre sua definição em 3 V's, ou três características principais: 1) volume, que diz respeito à quantidade de dados produzidos, 2) variedade, que demonstra a diversidade de tipos desses dados e 3) velocidade, que se refere à rapidez com que novos dados são gerados (ELGENDY; ELRAGAL, 2014; DIETRICH; HELLER; YANG, 2015; CIELEN; MEYSMAN; ALI, 2016; REIN; MEMMERT, 2016; PRAVEENA; BHARATHI, 2017; SKIENA, 2017; KAUR; JAGDEV, 2020). Alguns autores também consideram a existência de outras características que definem o *Big Data*, dos quais podemos destacar a veracidade, que se refere a confiabilidade desses dados, ou o grau de precisão que eles possuem (PRAVEENA; BHARATHI, 2017; VASSAKIS; PETRAKIS; KOPANAKIS, 2018; KAUR; JAGDEV, 2020).

Os dados produzidos diariamente possuem diversas formas, podendo ser classificados como a) estruturados, aqueles que possuem tipo, formato e estrutura definidos, e b) não estruturados, dados que não possuem forma definida e que não podemos organizar em uma tabela, como por exemplo, uma imagem ou vídeo (DIETRICH; HELLER; YANG, 2015; CIELEN; MEYSMAN; ALI, 2016). Adicionalmente, Dietrich, Heller e Yang (2015) ainda apresentam outras duas formas de definir os tipos de dados, sendo eles dados que não são se encaixam como estruturados ou não estruturados: a) semiestruturados, dados textuais que possuem padrões discerníveis e que permitem a análise, e b) *quasi* estruturado, dados textuais com formato não definido, mas que podem ser estruturados através de ferramentas.

As diferentes características que compõem o *Big Data*, somado às diferentes formas que os dados podem se apresentar, geram desafios em todas as etapas do manuseio dos dados, desde a coleta, até a visualização. O método de investigar dados a fim de encontrar conhecimentos úteis, através de um processo de descoberta ou de elaboração de hipóteses e testes dessas hipóteses, é chamado de *Data Science* (NIST, 2015). Trata-se de um campo ainda em

desenvolvimento, mas que pode ser definido como uma área que requer uma abordagem multidisciplinar sobre os dados, e que possui uma relação muito forte tanto com *Big Data*, quanto com tecnologias orientadas a dados, que causam mudanças na área da pesquisa e da indústria (DEMCHENKO et al., 2016).

No âmbito esportivo é comum encontrar trabalhos que utilizam uma "subárea" de *Data Science*, chamada *Data Mining*. Como explicam Han, Kamber e Pei (2012), o termo *Data Mining* talvez não expresse todas as etapas envolvidas no seu processo ou transpareça seu real objetivo, uma vez que ao aplicar *Data Mining* a uma base de dados, buscamos extrair conhecimento; por essa razão, outro nome que é comumente atribuído ao processo é o *Knowledge Discovery from Data* (Descoberta de conhecimento a partir dos dados). Como uma definição curta de *Data Mining*, podemos considerá-lo como um processo interativo de descoberta de padrões, de forma automatizada ou manual, a partir de dados, usando matemática, estatística e ferramentas de programação (HAN; KAMBER; PEI, 2012; NIST, 2015; ZHANG, 2017; KANTARDZIC, 2020).

Data Mining no cenário esportivo, pode ser definido como o processo no qual se utiliza modelos matemáticos ou estatísticos sofisticados para extrair informações valiosas, válidas e que sejam aplicáveis, a partir de uma base de dados, com o objetivo de ajudar e direcionar os atletas para uma melhor performance esportiva (DONG; CALVO, 2007). Entretanto, o uso dos resultados obtidos através dessas técnicas podem ser úteis para os vários públicos envolvidos com o esporte, e não só para os atletas, desde os técnicos e dirigentes, até os fãs e a mídia (VINUÉ, 2020). Predição de resultados através da análise de estatísticas de jogo e relação entre indicadores de performance e o resultado da partida (HAGHIGHAT; RASTEGARI; NOURAFZA, 2013; ZIMMERMANN, 2016; THABTAH; ZHANG; ABDELHAMID, 2019) e visualização de dados (VINUÉ, 2020), são algumas das possibilidades de utilização do método no esporte.

Apesar da presença de estudos na área, ainda existem muitas lacunas a serem preenchidas, como a melhor investigação na predição de performance, utilizando outras variáveis no modelo, modelos de detecção de talentos mais robustos, o desenvolvimento de ferramentas que unifiquem essas informações em um só local, tornando esses dados disponíveis e abertos para uma quantidade maior interessados no seu uso (BAI; BAI, 2021). Além disso o uso de conhecimentos em programação pode ser visto na coleta de dados de bases abertas como do *site* de campeonatos dos mais diversos esportes que, normalmente é realizado de forma manual, a qual é sujeita a erros, além de ser um processo extremamente maçante e demorado (JONNALAGADDA; GOYAL; HUFFMAN, 2015), e que pode ser realizado em poucos

minutos através de um *script* escrito em *Python* (“A Practical Introduction to Web Scraping in Python – Real Python”, 2022).

Diante do exposto, o trabalho foi pautado no seguinte questionamento: como realizar o processo de *Data Mining* no âmbito esportivo, através do uso de programação? A escolha da linguagem de programação *Python*, para a realização desse estudo, se deu pela facilidade do seu uso, familiaridade do autor com a ferramenta, e pelo diversos usos possíveis que a linguagem possui através das mais de 300 mil bibliotecas disponíveis (“PyPI · The Python Package Index”, 2021). Os dados utilizados ao longo do trabalho, se referem a arremessos de quadra de basquetebol, mas é importante ressaltar que o intuito do trabalho é apresentar o processo de *Data Mining* dentro do âmbito esportivo e não fazer inferências sobre a importância dos arremessos de quadra dentro do jogo de basquetebol. Entretanto, esse trabalho pode auxiliar os pesquisadores em futuros estudos dentro dessa temática.

1.1 OBJETIVOS

Nessa seção são apresentados os objetivos geral e específicos, que norteiam a elaboração desse trabalho.

1.1.1 Objetivo Geral

- Demonstrar um passo a passo de parte do processo de Data Mining no âmbito esportivo, utilizando programação.

1.1.2 Objetivos Específicos

- Propor o uso de ferramentas de programação para coleta e visualização de dados no âmbito esportivo;
- Coletar e organizar dados do site da NBA para utilização em futuros trabalhos;

1.2 JUSTIFICATIVA

A cada ano que passa, a quantidade de dados disponíveis sobre os mais diversos assuntos só aumenta. Seja na área da saúde, como no combate contra o SARS-COV-2 (QUOC-VIET

PHAM et al., 2020), na pesquisa de mercado ao analisar tendências em redes sociais (CHEN; CHIANG; STOREY, 2012; GANDOMI; HAIDER, 2015) ou na área esportiva (SCHELLING; ROBERTSON, 2020; LI; WANG; LI, 2021; WATANABE; SHAPIRO; DRAYER, 2021), dados ocupam um importante papel na descoberta de conhecimentos e na tomada de decisão.

Em uma menor escala, a facilidade provida pelas ferramentas de programação em todo processo de pesquisa, pode beneficiar em muito a vida do pesquisador, desde a coleta, passando pela análise desses dados, até as representações gráficas das descobertas realizadas. Apesar disso e do crescente número de publicações utilizando esporte associado ao *Data Mining*, o estímulo ao graduando de educação física na busca por essas novas ferramentas é quase inexistente. Ao olharmos para o currículo do curso na Universidade Federal de Santa Catarina, apenas uma disciplina aproxima-se do assunto: Introdução à Estatística, disponibilizada na 6ª fase. Em um mundo cada vez mais dependente da tecnologia, a aproximação da graduação com novos recursos se faz necessária.

Além disso, o interesse para a elaboração desse trabalho partiu do gosto do autor por programação, somado ao gosto pelo basquetebol. Acredita-se que se pode desenvolver um conhecimento muito mais aprofundado sobre os esportes, quando olhamos para os dados que eles geram, podendo-se exercer a função de treinador ou analista de forma muito mais satisfatória e embasada.

Com base no exposto, esse trabalho foi pensado como uma forma de divulgar esses conhecimentos, e encorajar outros da área da Educação Física, a aprenderem e aplicarem essas técnicas em suas pesquisas. Os principais beneficiados pelo conteúdo do trabalho, serão os pesquisadores, que desejam incrementar mais ainda suas pesquisas, e estudantes que visam uma carreira como analista esportivo.

2 REVISÃO DE LITERATURA

2.1 BIG DATA E DATA MINING

Nessa seção da revisão de literatura, serão abordados os temas *Big Data*, para que haja uma contextualização sobre o tema, apesar de não ser o foco principal do trabalho, e *Data Mining*, explicando cada passo do processo. Entretanto, para que se possa falar desses temas, é necessária uma explanação breve sobre uma variável presente em ambos, por essa razão, o primeiro tópico dessa revisão se encarregará de explicar o que são dados.

2.1.1 Dados

Dados podem ser definidos como o conhecimento que se tem sobre algo, e que pode ser usado para solucionar uma questão, ou uma informação que identifica algo ou alguém ("dados - significado de Dados", 2021). Eles podem representar valores tanto qualitativos, quanto quantitativos (ZHANG, 2017), e ser classificados a partir de sua forma. A previsão é de que até 2025, o volume de dados existentes será de 150 *zettabytes* (10^{21} *bytes* ou 150 trilhões de *gigabytes*) (KULKARNI, 2019). As fontes desses dados são diversas, desde *posts* em redes sociais como *Facebook* e *Instagram*, até dados de geolocalização e sequenciamento genético (DIETRICH; HELLER; YANG, 2015).

É comum observarmos na literatura dois grandes grupos para separação dos tipos de dados, os estruturados e os não estruturados. Os dados estruturados são caracterizados por possuir tipo, formato e estrutura definidos, de tal forma que seja possível avaliar as informações contidas de forma mais fácil. É atribuída aos dados estruturados a característica de estar contido em um modelo ou formato pré-definido, como uma planilha, por exemplo, possuindo rótulos para identificar cada coluna separadamente. Enquanto os dados não estruturados são aqueles que não possuem forma definida e não podemos organizá-lo com facilidade em forma de tabelas, por seu conteúdo ser específico do contexto ou por ser variável, como por exemplo vídeos e imagens (DIETRICH; HELLER; YANG, 2015; CIELEN; MEYSMAN; ALI, 2016; ERL; KHATTAK; BUHLER, 2016; SKIENA, 2017; KANTARDZIC, 2020). Da totalidade dos dados disponíveis hoje em dia, existe uma proporção muito maior de dados não estruturados, ao passo que sua taxa de crescimento é maior do que a de dados estruturados. Apesar da grande

disponibilidade de dados não estruturados, eles são mais difíceis de serem trabalhados, por requererem técnicas mais sofisticadas de análise (ERL; KHATTAK; BUHLER, 2016).

É comum encontrar outras divisões nos tipos de dados, como os dados semiestruturados, que se trata de dados textuais que possuem padrões discerníveis e que podem ser analisados, como arquivos XML (eXtensible Markup Language) (DIETRICH; HELLER; YANG, 2015; ERL; KHATTAK; BUHLER, 2016), e os dados *quasi* estruturados, dados de texto que com formato errático mas que pode ser formatado após muito trabalho (DIETRICH; HELLER; YANG, 2015). A Tabela 1 apresenta alguns exemplos dos tipos de dados estruturados e não estruturados.

Tabela 1: Tipos de dados

Estruturado	Não estruturado
Arquivos CSV	Vídeos
Planilhas eletrônicas	Fotos
Bancos de dados	E-mails
Arquivos XML	Áudio
Arquivos JSON	Arquivos de PDF

Fonte: elaborado pelo autor.

2.1.2 Big Data

Apesar do presente trabalho não tratar de conjuntos de dados muito grandes, se faz necessário uma breve explicação sobre os temas *Big Data* e *Data Science*, uma vez que uma das justificativas para a exploração dessa área tratada aqui nesse estudo, se basear na pouca divulgação dos temas de tecnologia no âmbito esportivo, dentro da graduação de educação física. Portanto, ao expor as definições para os termos citados acima, a intenção é instigar a curiosidade para o assunto, a fim de incentivar pesquisas mais aprofundadas nesse campo.

O termo *Big Data* é dado a conjuntos de dados muitos grandes, nos quais as vias convencionais de análise se tornam pouco efetivas (ELGENDY; ELRAGAL, 2014; CIELEN; MEYSMAN; ALI, 2016). Essa definição suscinta, entretanto, não é consenso e, normalmente o termo é descrito a partir de suas características, os três V's. sendo a primeira característica atribuída a ele, o volume, que diz respeito ao tamanho ou quantidade de dados. O volume dos dados no contexto de *Big Data* pode ser facilmente percebido, pois aqui, deixamos de usar planilhas para armazenar os dados, precisando de cada vez mais espaço. (ELGENDY;

ELRAGAL, 2014; DIETRICH; HELLER; YANG, 2015; CIELEN; MEYSMAN; ALI, 2016; REIN; MEMMERT, 2016; PRAVEENA; BHARATHI, 2017; SKIENA, 2017; KAUR; JAGDEV, 2020).

A segunda característica é a variedade, que demonstra a diversidade desses dados, uma vez que as fontes desses dados são das mais diversas, desde dados referentes a compras em *sites* de *e-commerce*, assuntos mais falados em redes sociais, até dados de geolocalização e, a partir de cada fonte, os dados podem assumir formas distintas. O terceiro V, é a velocidade, que se refere à rapidez com que novos dados são gerados a todo instante, e devolvidos a partir dos conjuntos analisados. Podemos usar como exemplo a geração de dados a partir de eventos esportivos e culturais transmitidos ao vivo, ou da reação do público sobre esses eventos através de mídias sociais, dados esses que são gerados em tempo real e precisam ser tratados na mesma rapidez (ELGENDY; ELRAGAL, 2014; DIETRICH; HELLER; YANG, 2015; CIELEN; MEYSMAN; ALI, 2016; REIN; MEMMERT, 2016; PRAVEENA; BHARATHI, 2017; SKIENA, 2017; KAUR; JAGDEV, 2020).

Alguns autores consideram a existência de outras características para o *Big Data*, abaixo estão as principais:

- Veracidade: que diz respeito à confiabilidade desses dados, ou o grau de precisão que eles possuem, podem definir se os dados são “bons” ou “ruins” para análise (PRAVEENA; BHARATHI, 2017; VASSAKIS; PETRAKIS; KOPANAKIS, 2018; KAUR; JAGDEV, 2020).
- Variabilidade: que diz respeito aos diferentes sentidos ou significados que um mesmo dado pode ter, dependendo do contexto, tornando difícil os processos de manuseio e gerenciamento (PRAVEENA; BHARATHI, 2017; VASSAKIS; PETRAKIS; KOPANAKIS, 2018).
- Valor: que diz respeito ao quanto esses dados podem carregar de valor dependendo do fim para o qual será utilizado. É uma característica que pode variar, dependendo da pessoa ou organização que olha para aquele conjunto de dados e define se ele é ou não valioso para o seu uso (KAUR; JAGDEV, 2020).

Levando em consideração todas as características que compõem o *Big* Data, tem-se um cenário onde todas as etapas de tratamento desses dados apresentam desafios, desde a coleta, até a visualização. A solução para esses desafios partem de uma área que demanda uma abordagem multidisciplinar sobre os dados, chamada *Data Science* (DEMCHENKO et al.,

2016). O termo pode ser definido como o método de investigar dados a fim de encontrar conhecimentos úteis, através de um processo de descoberta ou de elaboração de hipóteses e testes dessas hipóteses (NIST, 2015). A quantidade cada vez maior de dados disponíveis, leva a uma crescente necessidade para análise e utilização de forma concreta dessas informações. A relevância e importância do profissionais que trabalham nessa área é cada vez mais reconhecida e por essa razão, a área de *Data Science* tem cada vez mais crescido no mercado de trabalho (DIGITAL, 2021; G1 - O PORTAL DE NOTÍCIAS DA GLOBO, 2021).

2.1.3 Data Mining

Data Mining pode ser definido como um processo interativo de descoberta de padrões, de forma automatizada ou manual, a partir de dados, usando matemática, estatística e ferramentas de programação (HAN; KAMBER; PEI, 2012; NIST, 2015; ZHANG, 2017; KANTARDZIC, 2020). Han, Kamber e Pei (2012), questionam se o nome *Data Mining*, é o mais adequado para definir um processo tão complexo, uma vez que, fazendo uma analogia com a mineração de metais, quando se minera ouro a partir de pedras e areia, nós usamos apenas a expressão minerando ouro, excluindo as pedras e areia da equação, por não se tratar do objetivo primário da exploração. Assim, o termo talvez não expresse todas as etapas envolvidas no seu processo ou transpareça seu real objetivo, uma vez que ao aplicar *Data Mining* a uma base de dados, como na mineração de ouro, o que se busca extrair trata-se de um objeto em específico, o conhecimento. Por essa razão, o processo é muitas vezes chamado também de *Knowledge Discovery from Data* (Descoberta de conhecimento a partir dos dados), apesar de se tratar de processos um pouco distintos.

Kantardzic (2020) explica que o processo de *Data Mining* não é direto ao ponto, e dificilmente uma única aplicação de métodos resolva o problema, pelo contrário, trata-se de um processo que exige interatividade entre suas etapas, onde normalmente volta-se a etapas anteriores para se obter melhora em fases posteriores. As etapas apresentadas por Kantardzic (2020), que compõem todo o processo de *Data Mining* pode ser observado na Tabela 2.

Tabela 2: Etapas do processo de *Data Mining* segundo Kantardzic (2020).

Declarar o problema e formular a hipótese	Nessa etapa é necessário um conhecimento sobre o tipo de dado com o qual está se trabalhando, para que as hipóteses formuladas tenham sentido.
Coletar os dados	Retirar os dados das fontes e armazenar em arquivos que possam ser trabalhados.
Fazer o pré-processamento dos dados	Realizar a "limpeza" desses dados, tornando-os mais fáceis de serem entendidos e utilizados pelo modelo.
Estimar o modelo	Estudar qual técnica é a mais adequada para solucionar o problema declarado na primeira etapa. Aqui é normal voltar para etapas anteriores para se obter o melhor resultado.
Interpretar o modelo e apresentar conclusões	Entender o que o modelo aplicado nos traz de informação e de que forma essa informação pode ser útil na tomada de decisão .

Fonte: adaptado de Kantardzic (2020).

2.2 DATA MINING NO ESPORTE

Nessa seção da revisão, serão apresentadas visões sobre o Data Mining no esporte, lacunas a serem preenchidas e trabalhos que utilizaram o processo.

No esporte, o Data Mining pode ser definido como o processo pelo qual se utiliza modelos matemáticos ou estatísticos sofisticados para extrair informações valiosas, válidas e que sejam aplicáveis, a partir de uma base de dados, com o objetivo de ajudar e direcionar os atletas para uma melhor performance esportiva (DONG; CALVO, 2007). Vinué (2020), porém, destaca que o acesso a informações obtidas por esse meio, pode ser útil não só para os atletas, mas para todos os públicos que permeiam o contexto esportivo, desde técnicos e dirigentes, até os fãs e a mídia. Talvez um dos usos mais comuns do Data Mining no contexto esportivo, seja o de predição de resultados através de análise de estatísticas de jogo. Ao se pesquisar o assunto

no internet, facilmente se encontra tutoriais para a elaboração de modelos e, isso se deve ao fato do crescente número de apostadores em sites de apostas (GLOBO, 2021).

No contexto científico o assunto também é abordado, como pode se observar na revisão de Haghighat, Rastegari e Nourafza (2013), que avaliou nove trabalhos que utilizaram Data Mining em seus métodos, desde sua coleta de dados e escolha dos recursos para o modelo, até a técnica de classificação selecionada. Os principais pontos destacados como desvantagens dos estudos avaliados estão relacionados com o tamanho da base de dados analisada e a baixa precisão dos modelos aplicados.

Zimmermann (2016), teve como objetivo explorar as diferenças e similaridades entre os modelos de predição de resultado aplicados na NCAAB (National Collegiate Athletic Association – Basketball, liga estadunidense de basquetebol universitário) e na NBA (National Basketball Association). Após a aplicação dos métodos e exploração de ambas as bases, o autor observou que o indicador "adjusted efficiencies" pode ser transferido dos modelos da NCAAB para os modelos da NBA, mas outros indicadores se mostraram pouco eficientes. As principais diferenças observadas, que fazem com que os modelos sejam distintos são: o número menor de partidas no universitário, um espectro mais amplo de habilidade durante a temporada regular e diferenças muito grandes entre as pós-temporadas das ligas. Seguindo na mesma área, Thabtah, Zhang e Abdelhamid (2019), tiveram como objetivo elaborar uma estrutura de Machine Learning para a predição de resultados de jogos da NBA, a partir de dados históricos (jogos que já aconteceram), e identificar quais indicadores exercem uma maior influência no resultado da partida. Após a elaboração dos modelos, obteve-se como resultado que um dos indicadores que mais influenciavam o resultado eram o rebotes defensivos. Embora o trabalho tenha encontrado certos achados, a predição de resultado através de modelos é um assunto muito controverso e deve-se ter cuidado ao transferir os resultados para o contexto prático.

Entrando no campo de transferência dos achados para a prática, Vinué (2020) apresenta uma solução para aproximar dos treinadores a tomada de decisão a partir dos dados, ao passo que permite à mídia e aos fãs um entendimento melhor sobre o jogo e suas particularidades. Em seu trabalho, o autor desenvolveu uma aplicação web que permite a visualização de diversos indicadores de desempenho de jogadores das competições europeias: Eurocup e Euroleague, e da liga espanhola ACB.

Apesar da presença de estudos na área, ainda existem muitas lacunas a serem preenchidas. Dessas, pode-se destacar a 1) melhor investigação na predição de performance, utilizando outras variáveis no modelo, 2) modelos de detecção de talentos mais robustos, 3) o desenvolvimento de ferramentas que unifiquem essas informações em um só local, tornando

esses dados disponíveis e abertos para uma quantidade maior de interessados no seu uso, e 4) a segurança dos dados, uma vez que eles são gerados por indivíduos, e os resultados obtidos a partir deles podem gerar impactos na vida de atletas e treinadores (BAI; BAI, 2021).

3 MÉTODOS

3.1 DELIMITAÇÃO DO ESTUDO

Esse trabalho será estruturado a partir de um passo a passo de um método, no qual se extrai conhecimento a partir dos dados, realizando uma análise desse exemplo a fim de estimular a compreensão do leitor, portanto, quanto aos objetivos da pesquisa, podemos classificá-la como uma pesquisa exploratória (GIL, 2009). Além disso, se trata de uma pesquisa aplicada, na qual se aplica conhecimentos pré-existentes, na tentativa de solucionar problemas específicos, e quantitativa, pois considera tudo que é quantificável e traduz os números em opiniões e informações, para assim classificá-las e analisá-las (SILVA; MENEZES, 2001). Quanto aos seus procedimentos, por se tratar de uma pesquisa que se propõe a utilizar uma determinada base de dados ao processo de *Data Mining*, a fim de gerar um guia de como realizá-lo, não é possível caracterizá-lo de forma direta nos tipos de estudo propostos por Gil (2009), contudo, se considerarmos a base de dados como objeto de estudo, e o processo de *Data Mining* ao qual a base será submetida como o meio para a obtenção de respostas a partir desse objeto de estudo, é possível encaixar esse trabalho como um estudo de caso.

3.2 FERRAMENTAS

Para desenvolver todos os códigos presentes nesse trabalho, desde a coleta até a visualização dos dados, foi utilizada a linguagem de programação *Python*. Lançado por Guido van Rossum em 1991, cresceu a partir de uma linguagem chamada ABC, a qual facilitava a utilização de microcomputadores (tecnologia recente nos anos 1980) por cientistas não-especialistas, pragmática que permanece na linguagem *Python* (UNPINGCO, 2021; “The Python Language Reference”, 2022), que possui sintaxe expressiva, e que muitas vezes é comparado a um pseudocódigo (forma genérica de escrever o código usando linguagem nativa) executável (OLIPHANT, 2007), tornando mais simples o seu entendimento.

Programar em *Python* permite ao usuário explorar uma grande quantidade de possibilidades, e isso se deve muito ao fato de que a linguagem possui diversas bibliotecas, com uma variedade muito grande de funções. Abaixo, estão listadas as bibliotecas, não nativas, utilizadas nesse trabalho, com uma breve descrição, segundo o repositório mais famoso da linguagem, o pip ("PyPI · The Python Package Index", 2021):

- **Pandas:** pacote que fornece estruturas de dados rápidas, flexíveis e expressivas, desenvolvidas para lidar com dados relacionais ou classificados, de forma fácil e intuitiva.
- **Requests:** pacote que permite solicitações HTTP com facilidade; utilizado para conectar a sites.
- **NBA API:** biblioteca desenvolvida para retornar de forma mais facilitada dados presentes no site da NBA.
- **Pyodbc:** biblioteca que permite a conexão a banco de dados ODBC, permitindo manipular o banco de dados através do script em *Python*.
- **Streamlit:** framework que permite a criação de APPs de forma rápida, utilizando apenas *Python*, sem a necessidade de utilizar HTML.
- **Matplotlib:** biblioteca abrangente para criar visualizações estáticas, animadas e interativas em *Python*.
- **Seaborn:** biblioteca para fazer gráficos estatísticos em *Python*, integrado ao Pandas e Matplotlib.

Durante a fase de tratamento dos dados, foi utilizado SQL (*Structured Query Language*), a principal linguagem utilizada para gerenciar dados armazenados em sistemas de bancos de dados, a qual permite, através de comandos simples, realizar operações para selecionar, armazenar, modificar ou deletar dados. Assim como o *Python*, é relativamente fácil de entender SQL, uma vez que se trata de uma linguagem declarativa e utiliza a língua inglesa para declarar aquilo que precisa ser executado (SILVA; ALMEIDA; QUEIROZ, 2016).

3.3 PROCEDIMENTOS

A base de dados utilizada nesse trabalho é composta por dados de arremesso de jogadores de basquetebol, extraídos do *site* oficial da *National Basketball League* (NBA, http://www.nba.com/), das temporadas de 1996-1997 até 2020-2021. A coleta foi realizada de forma automatizada, utilizando um *script* feito em *Python* para a realização de *Web Scraping*.

Como a coleta de dados faz parte do passo a passo que esse trabalho se propõe a apresentar, ela será detalhada em uma seção posterior. A base coletada tem como principais informações: nome do jogador, nome da equipe, e localização dos arremessos de quadra tentados e acertados e o tempo de jogo no momento do arremesso, durante toda a temporada.

Data Mining é um processo de descoberta de padrões e de conhecimento a partir de um determinado conjunto dados. Tal processo, descrito em detalhes na Tabela 2, não é direto e rápido, e é comum voltar a uma etapa anterior para obter melhora em uma fase posterior (KANTARDZIC, 2020). Apesar de existirem cinco etapas envolvidas na descoberta do conhecimento, o estudo se limitou a apenas três: coleta de dados, tratamento dos dados, apresentação dos resultados (visualização dos dados). A decisão de se limitar a apenas três das etapas se deu por dois motivos: 1) falta de conhecimento do autor em estatística, o que limitaria as perguntas que poderiam ser feitas com base nos dados, e 2) tempo insuficiente para a realização de todas as etapas.

4 RESULTADOS

Nessa seção, serão apresentadas as etapas envolvidas no desenvolvimento do objetivo ao qual esse trabalho se propôs a atingir, partindo de a descoberta de uma lacuna a ser explorada, passando pela coleta e tratamento dos dados, até chegar à visualização em forma de *dashboard*. Ao longo dos próximos tópicos, serão apresentados todos os códigos utilizados e os motivos que levaram ao seu desenvolvimento. Entretanto, é preciso uma breve explicação sobre a base pública, de onde os dados foram coletados, para que se entenda o motivo da sua coleta.

A *National Basketball League* (NBA), possui uma gigantesca base de dados disponível no seu site (http://www.nba.com), com informações atuais e históricas sobre todos os jogadores que já passaram pela liga e até mesmo aqueles que nem foram *draftados*[1]. É possível encontrar dados sobre aspectos defensivos e ofensivos, desempenho em testes físicos, infrações, e uma infinidade de outras informações. Isso se deve ao fato de que dados, quando bem interpretados, geram informação que, em uma liga com alto valor financeiro e comercial, se convertem em aumento de performance e, consequentemente, dinheiro. Independentemente do esporte, a compreensão sobre a importância dos dados no jogo nas ligas estadunidenses não é recente,

[1] Jogadores *draftados* são aqueles escolhidos pelas franquias da NBA no evento chamado NBA Draft, realizado sempre antes do início da temporada regular, no qual as equipes podem escolher jogadores jovens, vindos do basquetebol universitário ou de ligas estrangeiras.

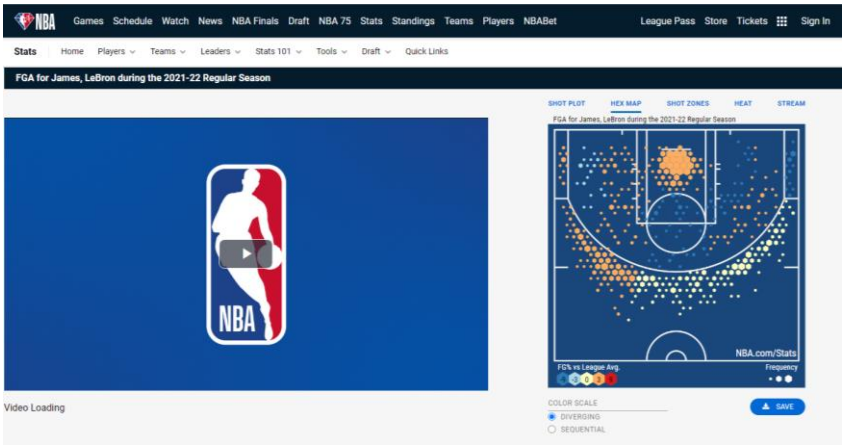
aspecto que fica evidente ao olharmos para os dados históricos presentes no site da NBA. É possível encontrar dados relevantes sobre todos os jogadores desde meados da década de 1990.

A quantidade muito grande de dados, impede que uma coleta manual seja realizada de forma satisfatória devido a sua propensão a erros e o tempo que se levaria para realizar o processo. Outro problema, é que a coleta desses dados muitas vezes é dificultada pelo próprio site, seja para evitar uma sobrecarga de acessos que levaria a queda do site, ou mesmo para evitar que aqueles dados (que custam dinheiro para serem gerados) sejam facilmente reutilizados.

4.1 COLETA DE DADOS

Entre todas as estatísticas disponíveis no site da liga, a visualização dos dados de lançamento dos jogadores chama a atenção (Figura 1). Nesse recurso é possível visualizar os dados de diferentes formas se tendo uma noção maior sobre os lançamentos naquela temporada. Entretanto, o recurso não permitia acessar informações históricas, mostrava apenas os lançamentos dos jogadores jogando aquela temporada, sendo assim, não permitia verificar a evolução do padrão de lançamento, ou fazer comparações entre diferentes temporadas de um mesmo jogador.

Figura 1: Dados de arremessos disponíveis no site da NBA.



Fonte: retirado de http://www.nba.com

Após uma breve pesquisa, foi verificado que os dados de outras temporadas existiam, só não estavam disponíveis de forma fácil no site. A partir dessa limitação, foi necessário explorar dentro do site, qual era a origem desses dados, e de que forma eles poderiam ser acessados. É natural que outras pessoas já tenham feito os mesmos questionamentos que nós e,

portanto, após uma breve busca na *Internet*, foi verificado que não só era possível, mas que outras pessoas já tinham feito o processo. Era necessário fazer a solicitação a uma URL específica do site da NBA, alterando alguns parâmetros para determinar o jogador e a temporada alvo, sendo também necessário utilizar cabeçalhos específicos para realizar a solicitação, usando uma analogia, a URL seria o nome da pessoa a qual você quer enviar uma carta, e os cabeçalhos são as informações que os Correios precisam para entregar a carta para a pessoa certa.

Depois de contornar o problema de solicitações repetidas ao site e, assim, possibilitar a coleta dos dados, outros questionamentos surgiram, como: é possível solicitar e mostrar os dados em tempo real? É possível solicitar os dados de diferentes jogadores ao mesmo tempo, para fazer uma comparação? Após alguns testes, as respostas foram surgindo: como forma de segurança contra ataques que sobrecarregam o site, a página da NBA não permite solicitações repetidas, levando a erros na execução do código e posteriormente a um bloqueio no acesso ao site, portanto, fazer solicitações repetidas ou elaborar o código para fazer solicitações constantes ao site da NBA era algo que precisava ser evitado. Pensando nisso, a solução foi manter todos esses dados de forma local, podendo acessar qualquer dado a qualquer momento, e quantas vezes fosse necessário.

4.1.1 Código de coleta dos dados

Abaixo, apresento a versão final do código para a coleta dos dados, dividindo-o em blocos, para que seja possível explicar cada etapa de sua construção:

Na Figura 2, pode ser observado o primeiro bloco de código. A primeira etapa na construção do código é importar as bibliotecas que serão utilizadas durante o programa. Nesse caso, foram importadas as bibliotecas: *requests, pandas, time, json, e nba_abi*. A biblioteca *requests*, tem a função de estabelecer conexão com o site da NBA; a biblioteca *pandas* permite o armazenamento da resposta dada pelo site em formato JSON para um *Dataframe*, uma estrutura de dados similar a uma tabela; a biblioteca *time*, nativa do *Python*, permite inserir o comando *sleep*, um tempo de pausa na execução do laço de repetição, evitando que o site da NBA derrube a conexão devido às múltiplas requisições; a biblioteca *json*, também nativa, permite manusear arquivos JSON; e a biblioteca *nba_api*, permite retornar o ID do time e dos jogadores, parâmetros necessários para compor a URL do site.

O segundo passo foi definir algumas funções que seriam reutilizadas durante o código. A *get_team_id()* tinha a objetivo de reunir o ID de todos os times presentes no site da NBA, *get_team_roster()* retornava o ID de todos os jogadores presentes no time e na temporada definidos, e a *get_players()* separava as informações recebidas pela *get_team_roster()* para uma estrutura de dicionários, possuindo ID e nome do jogador, que seriam utilizados posteriormente no código. Para finalizar, declarou-se o dicionário *teams_dict* para armazenar as informações obtidas pela função *get_team_id()*, chamada logo em seguida.

Figura 2: Primeiro bloco de código da coleta de dados

```
#Importar as bibliotecas que serão utilizadas ao longo do código
import requests as req
import pandas as pd
import time
import json
from nba_api.stats.endpoints.commonteamroster import CommonTeamRoster
from nba_api.stats.library.parameters import Season
from nba_api.stats.static import teams

#Definir as funções que serão reutilizadas ao longo do script, para evitar erros
def get_team_id():
    teams_nba = teams.get_teams()

    for team in teams_nba:
        id_team = team['id']
        fn_team = team['full_name']
        teams_dict[id_team] = fn_team

def get_season_roster(season, team_id):
    team_roster = CommonTeamRoster(season=season, team_id=team_id).get_data_frames()[0]
    return team_roster

def get_players(season, team_id):
    for index, row in get_season_roster(season, team_id).iterrows():
        fn_player = row['PLAYER']
        id_player = row['PLAYER_ID']
        players[id_player] = fn_player

teams_dict = {}
get_team_id()
```

Fonte: elaborado pelo autor.

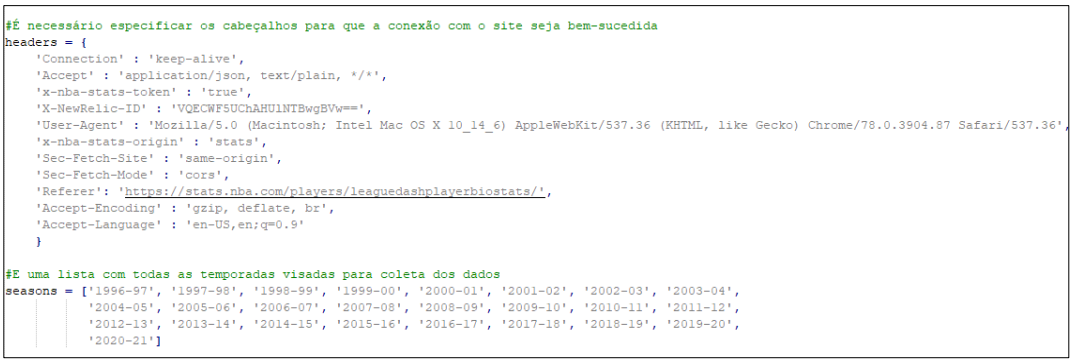
No segundo bloco de código (Figura 3), são declarados o dicionário *headers*, que armazena os cabeçalhos necessários para a conexão com o site da NBA, e a lista *seasons*, que possui as temporadas alvo para coleta dos dados.

No terceiro bloco de código (Figura 4), é iniciado um laço de repetição com o comando *for*, que vai percorrer todas as temporadas presentes na lista *seasons*, e mais um laço dentro que percorrerá o conteúdo do dicionário *teams_dict*, e um terceiro laço de repetição é colocado, para que percorra todos os jogadores daquele time. Em resumo, o código percorre todos os jogadores, de todos os times, em todas as temporadas. Dentro do terceiro laço de repetição é declarada a variável *url* que possui o endereço ao qual dever ser feita a requisição. Note que

estão em destaque *season* e *key_player*, que serão substituídos pelo valor do primeiro e do último laço de repetição, respectivamente (Figura 5).

Ainda na Figura 4, a conexão é estabelecida através do uso da biblioteca *requests*, e é realizada uma verificação quanto a resposta do site, para evitar que um erro ocorra. A resposta à requisição é armazenada na variável *r*, e em seguida é organizada em um *Dataframe* do *pandas*, para, por fim, ser exportado para um arquivo no formato CSV, com o seguinte padrão de nome: "temporada_nome do time_id do jogador.csv", por exemplo: "2020-21_Washington Wizards_1630264.csv".

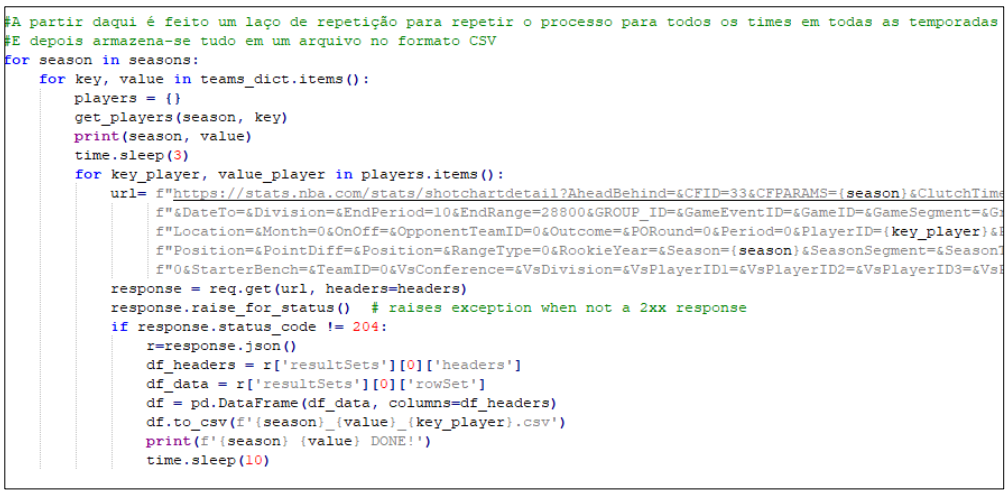
Figura 3: Segundo bloco de código da coleta de dados



```
#É necessário especificar os cabeçalhos para que a conexão com o site seja bem-sucedida
headers = {
    'Connection' : 'keep-alive',
    'Accept' : 'application/json, text/plain, */*',
    'x-nba-stats-token' : 'true',
    'X-NewRelic-ID' : 'VQECWF5UChAHUlNTBwgBVw==',
    'User-Agent' : 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.87 Safari/537.36',
    'x-nba-stats-origin' : 'stats',
    'Sec-Fetch-Site' : 'same-origin',
    'Sec-Fetch-Mode' : 'cors',
    'Referer': 'https://stats.nba.com/players/leaguedashplayerbiostats/',
    'Accept-Encoding' : 'gzip, deflate, br',
    'Accept-Language' : 'en-US,en;q=0.9'
    }

#E uma lista com todas as temporadas visadas para coleta dos dados
seasons = ['1996-97', '1997-98', '1998-99', '1999-00', '2000-01', '2001-02', '2002-03', '2003-04',
           '2004-05', '2005-06', '2006-07', '2007-08', '2008-09', '2009-10', '2010-11', '2011-12',
           '2012-13', '2013-14', '2014-15', '2015-16', '2016-17', '2017-18', '2018-19', '2019-20',
           '2020-21']
```

Fonte: elaborado pelo autor.

Figura 4: Terceiro bloco de código da coleta de dados



```
#A partir daqui é feito um laço de repetição para repetir o processo para todos os times em todas as temporadas
#E depois armazena-se tudo em um arquivo no formato CSV
for season in seasons:
    for key, value in teams_dict.items():
        players = {}
        get_players(season, key)
        print(season, value)
        time.sleep(3)
        for key_player, value_player in players.items():
            url= f"https://stats.nba.com/stats/shotchartdetail?AheadBehind=&CFID=33&CFPARAMS={season}&ClutchTime
                f"&DateTo=&Division=&EndPeriod=10&EndRange=28800&GROUP_ID=&GameEventID=&GameID=&GameSegment=&Gr
                f"Location=&Month=0&OnOff=&OpponentTeamID=0&Outcome=&PORound=0&Period=0&PlayerID={key_player}&P
                f"Position=&PointDiff=&Position=&RangeType=0&RookieYear=&Season={season}&SeasonSegment=&SeasonT
                f"0&StarterBench=&TeamID=0&VsConference=&VsDivision=&VsPlayerID1=&VsPlayerID2=&VsPlayerID3=&VsP
            response = req.get(url, headers=headers)
            response.raise_for_status()  # raises exception when not a 2xx response
            if response.status_code != 204:
                r=response.json()
                df_headers = r['resultSets'][0]['headers']
                df_data = r['resultSets'][0]['rowSet']
                df = pd.DataFrame(df_data, columns=df_headers)
                df.to_csv(f'{season}_{value}_{key_player}.csv')
                print(f'{season} {value} DONE!')
                time.sleep(10)
```

Fonte: elaborado pelo autor.

A decisão por não considerar a temporada atual (2021-22), se deve ao fato de que no momento da coleta, a temporada estava na metade e, portanto, os dados estariam incompletos. Vale destacar, que até chegar à versão final do código, foi necessário testar diversas vezes,

sendo que na maioria das vezes, o resultado do teste era uma mensagem de erro, portanto, o processo de *data mining*, em todas as suas etapas, não é direto ao ponto, é necessário ir e voltar no processo, para que se tenha o melhor resultado possível. O tempo de execução do código foi de aproximadamente 12 horas, sendo que a conexão com a internet, e o bloqueio do site da NBA devido às constantes requisições, apesar das medidas tomadas para minimizar esse problema, como a colocação de um "tempo de repouso" entre as solicitações, foram um limitador no processo, que precisou ser reiniciado algumas vezes. Como resultado, foram gerados 10657 arquivos CSV.

Figura 5: URL utilizada para acessar as informações

```
        url="https://stats.nba.com/stats/shotchartdetail?AheadBehind=&CFID=33&CFPA
RAMS={season}&ClutchTime=&Conference=&ContextFilter=&ContextMeasure=FGA&DateFrom=
&DateTo=&Division=&EndPeriod=10&EndRange=28800&GROUP_ID=&GameEventID=&GameID=&Gam
eSegment=&GroupID=&GroupMode=&GroupQuantity=5&LastNGames=0&LeagueID=00&Location=&
Month=0&OnOff=&OpponentTeamID=0&Outcome=&PORound=0&Period=0&PlayerID={key_player}
&PlayerID1=&PlayerID2=&PlayerID3=&PlayerID4=&PlayerID5=&PlayerPosition=&PointDiff
=&Position=&RangeType=0&RookieYear=&Season={season}&SeasonSegment=&SeasonType=Reg
ular+Season&ShotClockRange=&StartPeriod=1&StartRange=0&StarterBench=&TeamID=0&VsC
onference=&VsDivision=&VsPlayerID1=&VsPlayerID2=&VsPlayerID3=&VsPlayerID4=&VsPlay
                                erID5=&VsTeamID="
```

Fonte: elaborado pelo autor.

4.2 TRATAMENTO DE DADOS

O primeiro passo após coletar os dados, foi colocar todos os 10657 arquivos em um só, para que o manuseio dos dados se tornasse mais simples. Na Figura 6 está o código utilizado para unir todos os arquivos. Nessa fase, foram utilizadas as bibliotecas *os* e *pandas*, sendo a primeira nativa e com função de se comunicar com o Windows e identificar os arquivos presentes na pasta, e a segunda para armazenar e manusear os dados. A lista *file_list* armazena o nome de todos os arquivos que possuem a extensão CSV, e o laço de repetição passa por todos os arquivos presentes nessa lista, os lê e armazena em uma segunda lista, a *csv_list*. Essa, após o laço de repetição passar por todos os 10657 arquivos, é exportada novamente para CSV com o nome de "csv_merged.csv".

Ao unir todos os arquivos, outro problema surgiu: o arquivo final (csv_merged.csv) ficou grande demais (Figura 7) e tornava sua leitura pelo programa muito demorada. A alternativa foi migrar essa base de dados em formato CSV, para um formato em que fosse possível solicitar apenas os partes da base que seriam utilizados naquele momento, tornando o carregamento do programa muito mais rápido, e a utilização do *dashboard* muito mais

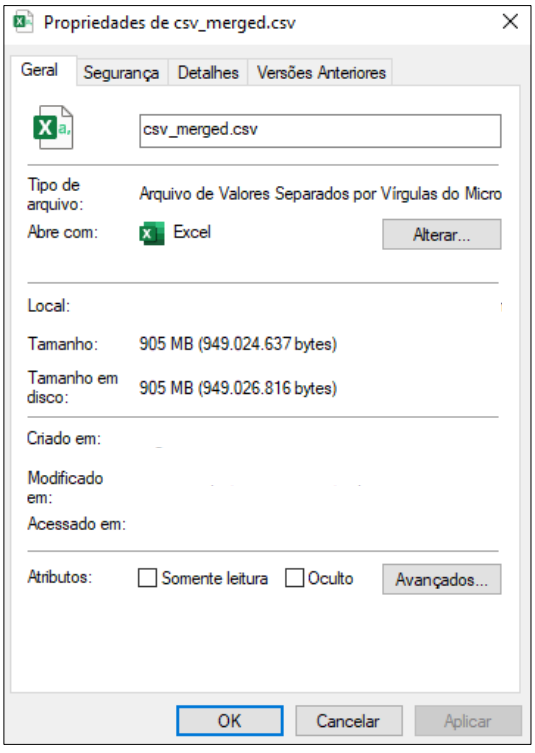
agradável para o usuário. Para fazer isso, foi criado um banco de dados em SQL, o que permitiu organizar o código para fazer consultas à base, sempre que necessário, trazendo informações precisas.

Figura 6: Unindo os arquivos CSV

```
#importando as bibliotecas
import os
import pandas as pd

#criando uma lista com todos os arquivos presentes na pasta atual
#que possuem a extensão CSV
file_list = [f for f in os.listdir() if f.endswith('.csv')]

#criando uma lista para armazenar as informações
csv_list = []

#laço de repetição para percorrer todos os arquivos, armazenar em um dataframe,
#incluir uma coluna de temporada e por fim armazenar em uma lista
for files in file_list:
    df = pd.read_csv(files, index_col=[0])
    df['SEASON'] = files[:4]
    csv_list.append(df)
print('FEITO!')

#coloca lista que possui todos os dados de todos os arquivos para um unico arquivo
csv_merged = pd.concat(csv_list, ignore_index=True)

csv_merged.to_csv('csv_merged.csv', index=False)
```

Fonte: elaborado pelo autor.

Figura 7: Tamanho do arquivo



Fonte: elaborado pelo autor.

A segunda etapa do tratamento dos dados, foi a criação do banco de dados em SQL. Para tal, foi utilizado o software Microsoft SQL Server Management Studio, e após a criação do banco de dados, a tabela para armazenar as informações foi criada utilizando o código exposto na Figura 8. A escolha do software em questão se deu por sua facilidade na criação de tabelas, uma vez que não é necessário conhecer a linguagem SQL, podendo-se criar tabelas apenas utilizando botões.

Figura 8: Criação da tabela em SQL

```
CREATE TABLE shot_data (
idshot_data int NOT NULL,
GRID_TYPE varchar(45) DEFAULT NULL,
GAME_ID int DEFAULT NULL,
GAME_EVENT_ID int DEFAULT NULL,
PLAYER_ID int DEFAULT NULL,
PLAYER_NAME varchar(45) DEFAULT NULL,
TEAM_ID int DEFAULT NULL,
TEAM_NAME varchar(45) DEFAULT NULL,
PERIO int DEFAULT NULL,
MINUTES_REMAINING int DEFAULT NULL,
SECONDS_REMAINING int DEFAULT NULL,
EVENT_TYPE varchar(45) DEFAULT NULL,
ACTION_TYPE varchar(45) DEFAULT NULL,
SHOT_TYPE varchar(45) DEFAULT NULL,
SHOT_ZONE_BASIC varchar(45) DEFAULT NULL,
SHOT_ZONE_AREA varchar(45) DEFAULT NULL,
SHOT_ZONE_RANGE varchar(45) DEFAULT NULL,
SHOT_DISTANCE varchar(45) DEFAULT NULL,
LOC_X int DEFAULT NULL,
LOC_Y int DEFAULT NULL,
SHOT_ATTEMPTED_FLAG varchar(45) DEFAULT NULL,
SHOT_MADE_FLAG varchar(45) DEFAULT NULL,
GAME_DATE varchar(45) DEFAULT NULL,
HTM varchar(45) DEFAULT NULL,
VTM varchar(45) DEFAULT NULL,
SEASON int DEFAULT NULL,
PRIMARY KEY (idshot_data)
)
```

Fonte: elaborado pelo autor.

A terceira etapa, foi incluir na tabela os dados presentes no arquivo *csv_merged.csv*. Para isso, foi elaborado um script que estabelecia a conexão com o banco de dados, recebia e lia o arquivo CSV e, em seguida, inseria os dados na tabela criada, nomeada de *shot_data*. As bibliotecas utilizadas nessa etapa foram a *pyodbc*, que realizava a conexão com o banco de dados, *csv* e *time*, bibliotecas nativas do *Python*, que tinham como função ler o arquivo CSV e adicionar um tempo de espera no programa, respectivamente.

Figura 9: Bloco 1 - Inserindo informações no banco de dados

```
#Declara as bibliotecas que serão utilizadas
import pyodbc
import csv
import time

#Estabele a conexão com o banco
dados_conexao = (
    "Driver={SQL Server};"
    "Server=NOME_DO_SERVIDOR;"
    "Database=shot_dashboard;"
)

conexao = pyodbc.connect(dados_conexao)

print('Conexão bem sucedida')

#Lê o arquivo com os dados de arremesso
with open('csv_merged.csv') as f:
    reader = csv.reader(f)
    data = list(reader)
print(f'\n {data[:4]} \n')

time.sleep(10)

#Elimina a primeira linha do arquivo, o cabeçalho
data.pop(0)
print(f'\n {data[:4]} \n')
```

Fonte: elaborado pelo autor.

Figura 10: Bloco 2 - Inserindo informações no banco de dados

```
cursor = conexao.cursor()

chave_primaria = 1

#Cria um laço de repetição, que passa por cada linha do arquivo e insere as informações em sua
#respectiva coluna na tabela shot_data do nosso banco de dados
for item in data:
    comando = f"""INSERT INTO shot_data(idshot_data, GRID_TYPE,
    GAME_ID, GAME_EVENT_ID, PLAYER_ID, PLAYER_NAME, TEAM_ID
    , TEAM_NAME, PERIO, MINUTES_REMAINING, SECONDS_REMAINING,
    EVENT_TYPE, ACTION_TYPE, SHOT_TYPE, SHOT_ZONE_BASIC
    , SHOT_ZONE_AREA, SHOT_ZONE_RANGE, SHOT_DISTANCE, LOC_X,
    LOC_Y, SHOT_ATTEMPTED_FLAG, SHOT_MADE_FLAG, GAME_DATE, HTM, VTM, SEASON)
VALUES ({chave_primaria}, '{item[0]}', {item[1]}, {item[2]}, {item[3]},
 '{item[4]}', {item[5]}, '{item[6]}', {item[7]}, {item[8]}, {item[9]},
 '{item[10]}', '{item[11]}', '{item[12]}', '{item[13]}', '{item[14]}',
 '{item[15]}', '{item[16]}', {item[17]}, {item[18]}, '{item[19]}',
 '{item[20]}', '{item[21]}', '{item[22]}', '{item[23]}', {item[24]})"""
    cursor.execute(comando)
    cursor.commit()
    print(f'Entrada número>>>{chave_primaria} --- Dados inseridos na tabela:'
          f'Jogador {item[4]}, Temporada {item[24]}')
    chave_primaria += 1


print('FECHOU!')
```

Fonte: elaborado pelo autor.

Na Figura 9, a primeira parte do código importa as bibliotecas que serão utilizadas, e logo em seguida, é feita a conexão com o banco de dados utilizando a biblioteca *pyodbc*. Após isso, é utilizado o comando *with* possibilitando a leitura do arquivo *csv_merged.csv*, que é então armazenado em lista, nomeada de *data*. Dado que o arquivo possui uma linha com os cabeçalhos de cada coluna, é necessário eliminá-lo, uma vez que esses cabeçalhos já estão presentes na nossa tabela em SQL; o comando *data.pop(0)*, realiza essa tarefa.

O próximo passo (Figura 10), é criar um cursor para que possamos passar as informações em SQL para o banco de dados, e criar uma variável *chave_primaria* que será utilizada para preencher a respectiva coluna na tabela. Em seguida é iniciado um laço de repetição que incluirá as informações lidas do arquivo *csv_merged.csv* para a tabela do banco de dados. A variável *comando* recebe o código em SQL, que possui tanto os cabeçalhos de cada coluna, quanto os dados que precisam ser incluídos nela. O comando *cursor.execute(comando)* executa o código em SQL no banco de dados, e o *cursor.commit()* garante que a inserção de dados seja validada. No fim de cada repetição, a variável *chave_primaria* é acrescida em 1, para que ela tenha um valor único para cada inserção no banco de dados.

4.3 VISUALIZAÇÃO DE DADOS

A última etapa na elaboração do trabalho é descobrir a melhor ou a forma mais adequada de mostrar os dados para o usuário (seja ele um cliente, treinador, atleta ou aluno). Essa fase tem grande importância no processo de Data Mining, uma vez que se não for possível demonstrar de forma clara as informações que os dados trazem, para o público geral, apenas um pequeno grupo de pessoas entenderá o que está apresentado. Mesmo tabelas, formas organizadas de agrupar os dados, quando muito grandes, como no caso da utilizada nesse trabalho, são muitos complexas de entender se não for aplicado algum outro tipo de visualização de dados sobre ela, como um gráfico de barras por exemplo.

A escolha do método foi influenciada por dois fatores principais: 1) era necessário elaborar uma forma de visualização que fosse interativa, na qual o usuário pudesse escolher qual informação gostaria de ver, e 2) dado o conhecimento limitado do autor sobre as ferramentas de desenvolvimento no *front-end*, seria interessante que a construção do gráfico fosse facilitada. Dessa forma, a ferramenta escolhida foi o framework de código aberto Streamlit, o qual permite, através de poucas linhas de código, elaborar aplicações bastante complexas e bonitas.

Diferente dos dois últimos tópicos do trabalho (Coleta dos dados e Tratamento dos dados), neste tratarei sobre o produto desenvolvido e, ao invés de trazer apenas os códigos utilizados, o foco será direcionado às funcionalidades e futuras possibilidades da ferramenta.

4.3.1 A solução

É importante, antes de focarmos na construção do dashboard interativo, ressaltar a impossibilidade de apresentar os dados em forma de tabela nessa situação. A base de dados coletada do site da NBA, em formato CSV, possuía mais de 4 milhões de linhas, organizadas em mais de 20 colunas, e muitos dos dados não faziam sentido quando olhados de forma separada, portanto era necessário amarrar informações para que se fosse possível entender as informações que a tabela contemplava.

Como os dados foram armazenados em um banco de dados, era necessário que o código do dashboard estabelecesse conexão com o banco para que pudesse ter acesso aos dados. Entretanto, a primeira etapa do código foi estabelecer algumas funções que seriam chamadas mais de uma vez pelo código. As funções no *Python*, funcionam como blocos de código, capazes de realizar alguma tarefa, como multiplicar 2 por 3, e que são armazenados com um determinado nome, como *funcao()*, para que seja reutilizado, sem a necessidade de repetir todo o bloco de código novamente. Essa prática leva você a escrever menos e, logo, diminui as chances de cometer erros de digitação ao longo do código. Essas funções, são declaradas no começo do código, após importar as bibliotecas, e possuem a seguinte sintaxe: "*def nome_da_funcao():*"; ao longo do código foram declaradas 11 funções (Figura 11), para retornar dados do banco de dados e elaborar o gráfico. Após declarar as funções, foi realizada a conexão com o banco de dados, utilizando novamente a biblioteca *pyodbc* e, a partir desse ponto a aplicação já era capaz de executar consultas a ele.

A construção do dashboard se deu sob a seguinte lógica: através de uma barra lateral, o usuário seria capaz de escolher a temporada alvo, o time dentro da temporada alvo e, por fim, o jogador dentro do time da temporada alvo. Conforme os filtros eram escolhidos, o programa consultava o banco e executava as funções necessárias para apresentar os resultados.

Como mencionado anteriormente, o ponto que mais despertou o meu interesse para realizar a coleta dessa base de dados, foram os dados referentes aos arremessos de quadra dos jogadores da NBA. Cada uma das linhas da tabela era referente a um arremesso realizado por algum jogador, durante alguma das temporadas abrangidas pela coleta, e cada coluna

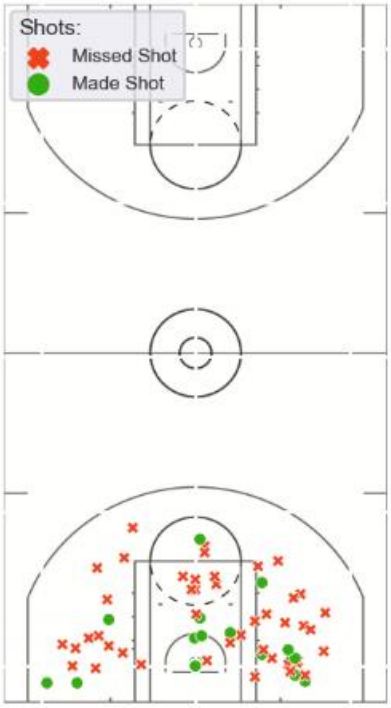
representava algum aspecto daquele arremesso, como: tempo de jogo, região da quadra, se o arremesso foi convertido em ponto ou não, e até mesmo a posição do arremesso, dado por duas informações *LOC_X* e *LOC_Y*, coordenadas cartesianas que poderiam ser plotadas em um gráfico. Essa última informação foi o ponto central de todo dashboard, plotar em forma de gráfico (Figura 12), todos os arremessos, de cada jogador, em cada temporada da NBA.

Figura 11: Funções

```
def season_select():
def team_name():
def player_name():
def shot_dados(season, team, player):
def plot_shot(dados):
def count_shot_att(season, team, player):
def count_shot_made(season, team, player):
def shot_made_area(season, team, player):
def shot_att_area(season, team, player):
def shot_miss_area(season, team, player):
def time_on_the_clock(season, team, player):
```

Fonte: elaborado pelo autor.

Figura 12: Gráfico de arremessos de quadra



Fonte: elaborado pelo autor.

Para elaborar a visualização, foi utilizado um gráfico de dispersão através da biblioteca *matplotlib* em conjunto com a *seaborn*, duas das mais utilizadas para elaboração de gráficos com *Python*. O gráfico de dispersão posicionou as coordenadas dadas pelas colunas *LOC_X* e *LOC_Y*, em um plano cartesiano que, com o auxílio da imagem de uma quadra de basquetebol como fundo do gráfico, permitia visualizar de forma aproximada a posição de cada arremesso de quadra tentado (convertidos e errados).

Ao lado do gráfico estão dispostas 12 métricas muito básicas (Tabela 3), retiradas também da base de dados, contemplando aspectos dos arremessos de cada jogador. Todas elas foram realizadas após a consulta ser feita ao banco, e em alguns casos, como na porcentagem de arremessos, foi necessário realizar um pequeno cálculo para retornar o resultado; no caso do *Best Zone*, foi utilizada a moda sobre a coluna, e nas métricas de porcentagem, foi realizado o cálculo de porcentagem, levando em consideração os arremessos tentados e convertidos. A

métrica *clutch_shots* representa a quantidade de arremessos acertados nos últimos dois minutos do último quarto do jogo.

O dashboard (Figura 13) torna o entendimento sobre os dados de forma mais facilitada, permitindo entender aquele grande amontoado de dados de forma visual, agrupando informações importantes.

Tabela 3: Métricas do Dashboard

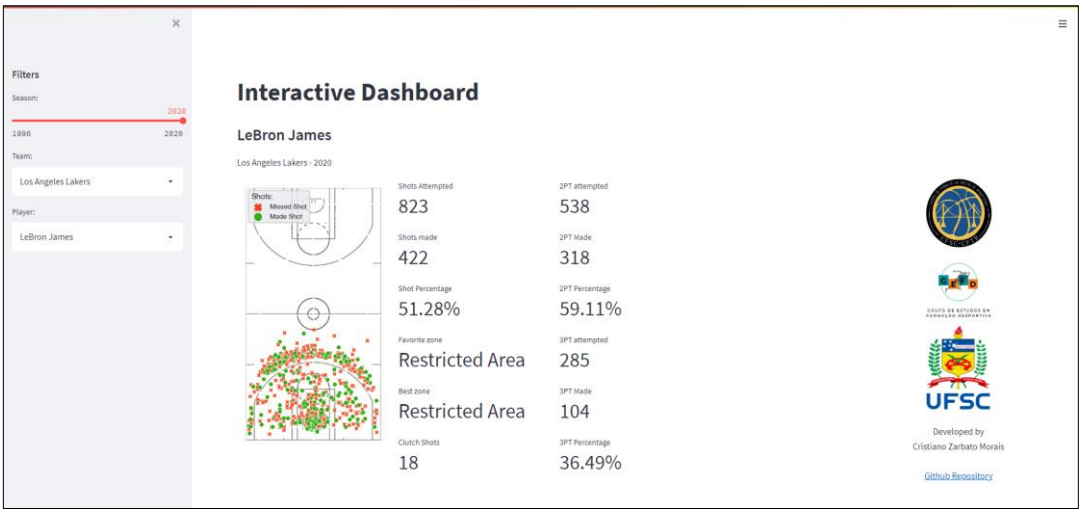
Shots Attempted	Arremessos totais tentados
Shots made	Arremessos convertidos
Shot Percentage	Porcentagem de Arremessos
Favorite zone	Zona da quadra com maior quantidade de arremessos
Best zone	Zona da quadra com maior quantidade de arremessos convertidos
Clutch shots	Arremessos convertidos nos últimos dois minutos do último quarto de jogo
2PT Attempted	Arremessos de 2 pontos tentados
2PT Made	Arremessos de 2 pontos convertidos
2PT Percentage	Porcentagem de arremessos de 2 pontos
3PT Attempted	Arremessos de 3 pontos tentados
3PT Made	Arremessos de 3 pontos convertidos
3PT Percentage	Porcentagem de arremessos de 3 pontos

Fonte: elaborado pelo autor.

4.3.2 Repositório

A fim de garantir a reprodutibilidade do trabalho e a constante melhoria do dashboard através da troca de experiências e conhecimentos, foi criado um repositório no GitHub, onde pode ser encontrado todos os códigos desenvolvidos em todas as etapas do trabalho, desde a coleta, até a construção do dashboard. Além disso, no mesmo repositório pode ser encontrado um *link* para o download da base de dados de forma direta. O repositório pode ser acessado em https://github.com/crizmorais/shot_dashboard.

Figura 13: Dashboard



Fonte: elaborado pelo autor.

5 DISCUSSÃO E CONCLUSÃO

O principal objetivo desse trabalho foi elaborar e apresentar um passo a passo de parte do processo de Data Mining no âmbito esportivo, utilizando programação. A partir da escolha dos dados que seriam utilizados, foram utilizados métodos de coleta, tratamento e por último visualização, utilizando as linguagens de programação *Python* e SQL. A transformação dos dados organizados em tabelas, em uma versão mais visual, como o dashboard proposto ao longo do trabalho, permite o melhor entendimento das informações que a base de dados carrega e, consequentemente, permite que possamos tirar conclusões mais acuradas sobre os dados.

Com o crescimento da área de dados, é importante ressaltar as facilidades e barreiras encontradas ao se coletar dados em bases públicas. É fato que, coletar os dados de forma manual tende ao erro, e se trata de um processo extremamente cansativo e maçante, no qual a coleta de bases muito grandes é impossibilitada tornando, consequentemente, as conclusões sobre os dados menos acuradas devido a pouca quantidade de informação. Ao se utilizar programação na coleta dos dados, é possível em poucos minutos coletar bases que demorariam horas de forma manual ou, outras que seriam inviáveis de serem coletadas, como é o caso da base utilizada nesse trabalho, que possui aproximadamente 4 milhões de linhas com mais de 20 colunas. Entretanto, é possível encontrar barreiras nesse processo, uma vez que os *sites* possuem mecanismos que impedem o acesso repetido de um usuário ao seu domínio, o que pode ser caracterizado como uma tentativa de derrubada do site. Além disso, uma vez que os dados

muitas vezes transmitem informações muito ricas, é comum que o site dificulte ou mesmo impeça sua coleta.

Ainda sobre a coleta de dados é preciso que, ao identificar a página de onde serão coletadas as informações, seja verificado de que forma os dados estão dispostos no site, uma vez que essa organização não segue um padrão. Para facilitar o processo, é recomendado que se se entenda ao menos o básico de HTML, para que se possa ler e entender o código por trás das páginas web. O processo de webscraping é bastante útil, mas pela necessidade de se adaptar a cada coleta, não segue um padrão bem definido, portanto é recomendado que aquele que tenha interesse em aprender, realize o processo diversas vezes para se tornar habituado, e assim conseguir aplicar à contextos mais complexos.

Na segunda etapa do trabalho, o tratamento de dados, a escolha de utilizar um banco de dados SQL, em um primeiro momento, pareceu difícil pela necessidade de aprender uma tecnologia diferente no meio do processo que já era bastante complexo. Entretanto, após algum tempo pesquisando, o SQL se mostrou bastante simples de compreender e extremamente útil e adequado para a tarefa que precisava ser realizada. Em adição, vale ressaltar que algumas melhorias no código são necessárias para que a integração entre dashboard e banco de dados seja mais veloz, e que outras informações possam ser adicionadas ao banco, a fim de tornar o produto mais completo, aspectos que serão explorados em trabalhos futuros.

Como comentado ao longo do trabalho, a ideia de uma visualização de dados de arremessos não é original, uma vez que o próprio site da NBA possui o recurso. O diferencial da solução proposta, é a possibilidade de trabalhar com dados históricos, e a liberdade para elaborar outros tipos de visualização e utilizar os dados para realizar análises de diferentes tipos. Além disso, é possível encontrar na literatura outros trabalhos que exploram a visualização de dados dentro do esporte e, visto a qualidade e a importância dos dados nesse contexto (PILEGGI et al., 2012; CHEN et al., 2016; WU et al., 2018; VINUÉ, 2020), é latente a necessidade de uma melhor exploração dessa área dentro da graduação em educação física.

A construção do presente trabalho se deu a partir de uma necessidade e curiosidade do autor sobre conhecimentos na área de programação. Entretanto, a maior parte dos métodos aplicados e descritos nos capítulos acima, foram buscados fora da graduação em educação física. Apesar da importância da tecnologia nos dias de hoje, não temos a aproximação do estudante de educação física com as possibilidades que o conhecimento em programação pode fornecer em sua formação, como demonstrado ao longo do trabalho .

Acredito que a ferramenta desenvolvida ao longo do trabalho, bem como os conhecimentos compartilhados ao longo dele, possam servir de base para que outros alunos de graduação desenvolvam seus próprios trabalhos, ou mesmo o interesse pela importância e necessidade do estudo de programação e tecnologias dentro da nossa área, pois enquanto não buscarmos o diferente, continuaremos atrás em conhecimento e relevância.

5.1 LIMITAÇÕES DO ESTUDO E FUTUROS ESTUDOS

Ao longo da elaboração desse trabalho, a etapa que apresentou maiores dificuldade foi a de coleta de dados. Uma vez que as empresas protegem seus sites contra ataques, solicitações repetidas são bloqueadas, e a coleta acaba se tornando mais demorada e dificultada. Outro problema ao se coletar os dados de forma automatizada é que, muitas vezes, a construção do site não se dá de forma bem estruturada, sendo mais difícil encontrar os padrões necessários para a automatização da tarefa via *script*.

Quanto a futuros estudos, como comentado em seções anteriores, é possível fazer outras perguntas aos dados, a fim de se tirar outras conclusões, como a comparação entre jogadores e entre temporadas para se tentar entender a evolução do arremesso dentro do jogo ao longo dos anos. Além disso, é possível extrair outros dados do site da NBA, para que a base seja complementada e as análises sobre os jogadores sejam mais completas. Pode-se, por exemplo, extrair as informações sobre rebotes defensivos e procurar relações entre essa estatística e os arremessos feitos em um contra-ataque. Por fim, com o intuito de complementar o dashboard elaborado, será adicionada a função de comparação, podendo-se visualizar dois ou mais jogadores simultaneamente.

REFERÊNCIAS

BAI, Z.; BAI, X. Sports Big Data: Management, Analysis, Applications, and Challenges. *Complexity*, v. 2021, p. 1–11, 30 jan. 2021.

CHEN; CHIANG; STOREY. Business Intelligence and Analytics: From Big Data to Big Impact. *MIS Quarterly*, v. 36, n. 4, p. 1165, 2012.

CIELEN, D.; MEYSMAN, A.; ALI, M. *Introducing data science: big data, machine learning, and more, using Python tools*. Shelter Island, NY: Manning Publications, 2016.

dados - significado de Dados. Disponível em: <https://www.dicio.com.br/dados/>. Acesso em: 24 ago. 2021.

DEMCHENKO, Y. et al. EDISON Data Science Framework: A Foundation for Building Data Science Profession for Research and Industry. Em: 2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Luxembourg, Luxembourg. *Anais*... Em: 2016 IEEE INTERNATIONAL CONFERENCE ON CLOUD COMPUTING TECHNOLOGY AND SCIENCE (CLOUDCOM). Luxembourg, Luxembourg: IEEE, dez. 2016. Disponível em: <http://ieeexplore.ieee.org/document/7830748/>. Acesso em: 14 ago. 2021.

DIETRICH, D.; HELLER, B.; YANG, B. (ed.). *Data science & big data analytics: discovering, analyzing, visualizing and presenting data*. Indianapolis, IN: Wiley, 2015.

DIGITAL, O. *Levantamento mostra aumento de mais de 450% nas vagas para cientistas de dados*. Disponível em: <https://olhardigital.com.br/2021/06/24/pro/levantamento-mostra-aumento-de-mais-de-450-nas-vagas-em-dados/>. Acesso em: 25 ago. 2021.

DONG, D.; CALVO, R. A. Integrating Data Mining Processes within the Web Environment for the Sports Community. Em: 2007 IEEE International Conference on Integration Technology, *Anais*... Em: 2007 IEEE INTERNATIONAL CONFERENCE ON INTEGRATION TECHNOLOGY. mar. 2007.

ELGENDY, N.; ELRAGAL, A. Big Data Analytics: A Literature Review Paper. Em: PERNER, P. (Ed.). *Advances in Data Mining. Applications and Theoretical Aspects*. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2014. 8557p. 214–227.

ERL, T.; KHATTAK, W.; BUHLER, P. *Big data fundamentals: concepts, drivers & techniques*. Boston : [Vancouver, BC]: Prentice Hall ; Service Tech Press, 2016.

G1 - O PORTAL DE NOTÍCIAS DA GLOBO. *Demanda por profissionais da área de dados cresce quase 500%; salários chegam a R$ 22 mil*. Disponível em: <https://g1.globo.com/economia/concursos-e-emprego/noticia/2021/07/05/demanda-por-profissionais-da-area-de-dados-cresce-quase-500percent-salarios-chegam-a-r-22-mil.ghtml>. Acesso em: 25 ago. 2021.

GANDOMI, A.; HAIDER, M. Beyond the Hype: Big Data Concepts, Methods, and Analytics. *International Journal of Information Management*, v. 35, n. 2, p. 137–144, abr. 2015.

GIL, A. C. *Como elaborar projetos de pesquisa*. São Paulo: Atlas, 2009.

GLOBO. *O mercado de apostas esportivasGente | Uma conexão Globo*, 6 ago. 2021. . Disponível em: <https://gente.globo.com/o-mercado-de-apostas-esportivas/>. Acesso em: 26 ago. 2021.

HAGHIGHAT, M.; RASTEGARI, H.; NOURAFZA, N. A Review of Data Mining Techniques for Result Prediction in Sports. v. 2, n. 5, p. 7, 2013.

HAN, J.; KAMBER, M.; PEI, J. *Data mining: concepts and techniques*. 3rd ed ed. Burlington, MA: Elsevier, 2012.

KANTARDZIC, M. *Data mining: concepts, models, methods, and algorithms*. Third edition ed. Piscataway, NJ: IEEE Press, 2020.

KAUR, G.; JAGDEV, G. Analyzing and Exploring the Impact of Big Data Analytics in Sports Science. Em: 2020 Indo – Taiwan 2nd International Conference on Computing, Analytics and Networks (Indo-Taiwan ICAN), Rajpura, Punjab, India. *Anais*... Em: 2020 INDO-TAIWAN 2ND INTERNATIONAL CONFERENCE ON COMPUTING, ANALYTICS AND NETWORKS (INDO-TAIWAN ICAN). Rajpura, Punjab, India: IEEE, fev. 2020. Disponível em: <https://ieeexplore.ieee.org/document/9181320/>. Acesso em: 29 jul. 2021.

KULKARNI, R. *Big Data Goes Big*. Disponível em: <https://www.forbes.com/sites/rkulkarni/2019/02/07/big-data-goes-big/>. Acesso em: 11 ago. 2021.

NIST. *NIST Big Data Interoperability Framework: Volume 1, Definitions*. [s.l.] National Institute of Standards and Technology, out. 2015. . Disponível em:

<https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.1500-1.pdf>. Acesso em: 14 ago. 2021.

OLIPHANT, T. E. Python for Scientific Computing. *Computing in Science & Engineering*, v. 9, n. 3, p. 10–20, 2007.

PRAVEENA, M. D. A.; BHARATHI, B. A survey paper on big data analytics. Em: 2017 International Conference on Information Communication and Embedded Systems (ICICES), Chennai, India. *Anais*... Em: 2017 INTERNATIONAL CONFERENCE ON INFORMATION COMMUNICATION AND EMBEDDED SYSTEMS (ICICES). Chennai, India: IEEE, fev. 2017. Disponível em: <http://ieeexplore.ieee.org/document/8070723/>. Acesso em: 29 jul. 2021.

PyPI · The Python Package Index. Disponível em: <https://pypi.org/>. Acesso em: 19 ago. 2021.

QUOC-VIET PHAM et al. Artificial Intelligence (AI) and Big Data for Coronavirus (COVID-19) Pandemic: A Survey on the State-of-the-Arts. 2020. Disponível em: <http://rgdoi.net/10.13140/RG.2.2.23518.38727>. Acesso em: 10 ago. 2021.

REIN, R.; MEMMERT, D. Big Data and Tactical Analysis in Elite Soccer: Future Challenges and Opportunities for Sports Science. *SpringerPlus*, v. 5, n. 1, p. 1410, dez. 2016.

SILVA, E. L. da; MENEZES, E. M. *Metodologia da Pesquisa e Elaboração de Dissertação*. 3. ed. [s.l: s.n.]

SILVA, Y. N.; ALMEIDA, I.; QUEIROZ, M. SQL: From Traditional Databases to Big Data. Em: Proceedings of the 47th ACM Technical Symposium on Computing Science Education - SIGCSE '16, Memphis, Tennessee, USA. *Anais*... Em: THE 47TH ACM TECHNICAL SYMPOSIUM. Memphis, Tennessee, USA: ACM Press, 2016. Disponível em: <http://dl.acm.org/citation.cfm?doid=2839509.2844560>. Acesso em: 8 jun. 2022.

SKIENA, S. S. *The Data Science Design Manual*. Cham: Springer International Publishing, 2017.

THABTAH, F.; ZHANG, L.; ABDELHAMID, N. NBA Game Result Prediction Using Feature Analysis and Machine Learning. *Annals of Data Science*, v. 6, n. 1, p. 103–116, mar. 2019.

The Python Language Reference. Disponível em: <https://docs.python.org/3/>.

UNPINGCO, J. *Python Programming for Data Analysis*. Cham: Springer International Publishing, 2021.

VASSAKIS, K.; PETRAKIS, E.; KOPANAKIS, I. Big Data Analytics: Applications, Prospects and Challenges. Em: SKOURLETOPOULOS, G. et al. (Ed.). *Mobile Big Data*. Lecture Notes on Data Engineering and Communications Technologies. Cham: Springer International Publishing, 2018. 10p. 3–20.

VINUÉ, G. A Web Application for Interactive Visualization of European Basketball Data. *Big Data*, v. 8, n. 1, p. 70–86, 1 fev. 2020.

ZHANG, A. *Data Analytics: Practical guide to leveraging the power of algorithms, data science, data mining, statistics, big data, and predictive analysis to improve business, work, and life*. [s.l: s.n.]

ZIMMERMANN, A. Basketball Predictions in the NCAAB and NBA: Similarities and Differences. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, v. 9, n. 5, p. 350–364, out. 2016.

APÊNDICE A – Coleta de dados

```
        #Importar as bibliotecas que serão utilizadas ao longo do código
        import requests as req
        import pandas as pd
        import time
        import json
        from nba_api.stats.endpoints.commonteamroster import CommonTeamRoster
        from nba_api.stats.library.parameters import Season
        from nba_api.stats.static import teams

        #Definir as funções que serão reutilizadas ao longo do script, para evitar
erros
        def get_team_id():
            teams_nba = teams.get_teams()

            for team in teams_nba:
                id_team = team['id']
                fn_team = team['full_name']
                teams_dict[id_team] = fn_team

        def get_season_roster(season, team_id):
            team_roster = CommonTeamRoster(season=season,
team_id=team_id).get_data_frames()[0]
            return team_roster

        def get_players(season, team_id):
            for index, row in get_season_roster(season, team_id).iterrows():
                fn_player = row['PLAYER']
                id_player = row['PLAYER_ID']
                players[id_player] = fn_player

        teams_dict = {}
        get_team_id()

        #É necessário especificar os cabeçalhos para que a conexão com o site seja
bem-sucedida
        headers = {
            'Connection' : 'keep-alive',
            'Accept' : 'application/json, text/plain, */*',
            'x-nba-stats-token' : 'true',
            'X-NewRelic-ID' : 'VQECWF5UChAHUlNTBwgBVw==',
            'User-Agent' : 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.87 Safari/537.36',
            'x-nba-stats-origin' : 'stats',
            'Sec-Fetch-Site' : 'same-origin',
            'Sec-Fetch-Mode' : 'cors',
            'Referer': 'https://stats.nba.com/players/leaguedashplayerbiostats/',
            'Accept-Encoding' : 'gzip, deflate, br',
            'Accept-Language' : 'en-US,en;q=0.9'
            }

        #E uma lista com todas as temporadas visadas para coleta dos dados
        seasons = ['1996-97', '1997-98', '1998-99', '1999-00', '2000-01', '2001-02',
'2002-03', '2003-04',
                   '2004-05', '2005-06', '2006-07', '2007-08', '2008-09', '2009-10',
'2010-11', '2011-12',
                   '2012-13', '2013-14', '2014-15', '2015-16', '2016-17', '2017-18',
'2018-19', '2019-20',
                   '2020-21']

        #A partir daqui é feito um laço de repetição para repetir o processo para
todos os times em todas as temporadas
        #E depois armazenar tudo em um arquivo no formato CSV
        for season in seasons:
            for key, value in teams_dict.items():
```

```
                players = {}
                get_players(season, key)
                print(season, value)
                time.sleep(3)
                for key_player, value_player in players.items():
                    url=
f"https://stats.nba.com/stats/shotchartdetail?AheadBehind=&CFID=33&CFPARAMS={season
}&ClutchTime=&Conference=&ContextFilter=&ContextMeasure=FGA&DateFrom=" \

f"&DateTo=&Division=&EndPeriod=10&EndRange=28800&GROUP_ID=&GameEventID=&GameID=&Gam
eSegment=&GroupID=&GroupMode=&GroupQuantity=5&LastNGames=0&LeagueID=00&" \

f"Location=&Month=0&OnOff=&OpponentTeamID=0&Outcome=&PORound=0&Period=0&PlayerID={k
ey_player}&PlayerID1=&PlayerID2=&PlayerID3=&PlayerID4=&PlayerID5=&Player" \

f"Position=&PointDiff=&Position=&RangeType=0&RookieYear=&Season={season}&SeasonSegm
ent=&SeasonType=Regular+Season&ShotClockRange=&StartPeriod=1&StartRange=" \

f"0&StarterBench=&TeamID=0&VsConference=&VsDivision=&VsPlayerID1=&VsPlayerID2=&VsPl
ayerID3=&VsPlayerID4=&VsPlayerID5=&VsTeamID="
                    response = req.get(url, headers=headers)
                    response.raise_for_status()  # raises exception when not a 2xx
response
                    if response.status_code != 204:
                        r=response.json()
                        df_headers = r['resultSets'][0]['headers']
                        df_data = r['resultSets'][0]['rowSet']
                        df = pd.DataFrame(df_data, columns=df_headers)
                        df.to_csv(f'{season}_{value}_{key_player}.csv')
                        print(f'{season} {value} DONE!')
                        time.sleep(10)
```

APÊNDICE B – Unindo arquivos

```
        #importando as bibliotecas
        import os
        import pandas as pd

        #criando uma lista com todos os arquivos presentes na pasta atual
        #que possuem a extensão CSV
        file_list = [f for f in os.listdir() if f.endswith('.csv')]

        #criando uma lista para armazenar as informações
        csv_list = []

        #laço de repetição para percorrer todos os arquivos, armazenar em um
dataframe,
        #incluir uma coluna de temporada e por fim armazenar em uma lista
        for files in file_list:
            df = pd.read_csv(files, index_col=[0])
            df['SEASON'] = files[:4]
            csv_list.append(df)
        print('FEITO!')

        #coloca lista que possui todos os dados de todos os arquivos para um unico
arquivo
        csv_merged = pd.concat(csv_list, ignore_index=True)

        csv_merged.to_csv('csv_merged.csv', index=False)
```

APÊNCIDE C – Criando tabela em SQL

```
CREATE TABLE shot_data (
  idshot_data int NOT NULL,
  GRID_TYPE varchar(45) DEFAULT NULL,
  GAME_ID int DEFAULT NULL,
  GAME_EVENT_ID int DEFAULT NULL,
  PLAYER_ID int DEFAULT NULL,
  PLAYER_NAME varchar(45) DEFAULT NULL,
  TEAM_ID int DEFAULT NULL,
  TEAM_NAME varchar(45) DEFAULT NULL,
  PERIO int DEFAULT NULL,
  MINUTES_REMAINING int DEFAULT NULL,
  SECONDS_REMAINING int DEFAULT NULL,
  EVENT_TYPE varchar(45) DEFAULT NULL,
  ACTION_TYPE varchar(45) DEFAULT NULL,
  SHOT_TYPE varchar(45) DEFAULT NULL,
  SHOT_ZONE_BASIC varchar(45) DEFAULT NULL,
  SHOT_ZONE_AREA varchar(45) DEFAULT NULL,
  SHOT_ZONE_RANGE varchar(45) DEFAULT NULL,
  SHOT_DISTANCE varchar(45) DEFAULT NULL,
  LOC_X int DEFAULT NULL,
  LOC_Y int DEFAULT NULL,
  SHOT_ATTEMPTED_FLAG varchar(45) DEFAULT NULL,
  SHOT_MADE_FLAG varchar(45) DEFAULT NULL,
  GAME_DATE varchar(45) DEFAULT NULL,
  HTM varchar(45) DEFAULT NULL,
  VTM varchar(45) DEFAULT NULL,
  SEASON int DEFAULT NULL,
  PRIMARY KEY (idshot_data)
)
```

APÊNDICE D – Incluindo dados na tabela

```
#Declara as bibliotecas que serão utilizadas
import pyodbc
import csv
import time

#Estabele a conexão com o banco
dados_conexao = (
    "Driver={SQL Server};"
    "Server=NOME_DO_SERVIDOR;"
    "Database=shot_dashboard;"
)

conexao = pyodbc.connect(dados_conexao)

print('Conexão bem sucedida')

#Lê o arquivo com os dados de arremesso
with open('csv_merged.csv') as f:
    reader = csv.reader(f)
    data = list(reader)
print(f'\n {data[:4]} \n')

time.sleep(10)

#Elimina a primeira linha do arquivo, o cabeçalho
data.pop(0)
print(f'\n {data[:4]} \n')

cursor = conexao.cursor()

chave_primaria = 1
```

```
        #Cria um laço de repetição, que passa por cada linha do arquivo e insere as
informações em sua
        #respectiva coluna na tabela shot_data do nosso banco de dados
        for item in data:
            comando = f"""INSERT INTO shot_data(idshot_data, GRID_TYPE,
            GAME_ID, GAME_EVENT_ID, PLAYER_ID, PLAYER_NAME, TEAM_ID
            , TEAM_NAME, PERIO, MINUTES_REMAINING, SECONDS_REMAINING,
            EVENT_TYPE, ACTION_TYPE, SHOT_TYPE, SHOT_ZONE_BASIC
            , SHOT_ZONE_AREA, SHOT_ZONE_RANGE, SHOT_DISTANCE, LOC_X,
            LOC_Y, SHOT_ATTEMPTED_FLAG, SHOT_MADE_FLAG, GAME_DATE, HTM, VTM, SEASON)
        VALUES ({chave_primaria}, '{item[0]}', {item[1]}, {item[2]}, {item[3]},
         '{item[4]}', {item[5]}, '{item[6]}', {item[7]}, {item[8]}, {item[9]},
         '{item[10]}', '{item[11]}', '{item[12]}', '{item[13]}', '{item[14]}',
         '{item[15]}', '{item[16]}', {item[17]}, {item[18]}, '{item[19]}',
         '{item[20]}', '{item[21]}', '{item[22]}', '{item[23]}', {item[24]})"""
            cursor.execute(comando)
            cursor.commit()
            print(f'Entrada número>>>{chave_primaria} --- Dados inseridos na
tabela:'
                  f'Jogador {item[4]}, Temporada {item[24]}')
            chave_primaria += 1


        print('FECHOU!')
```

APÊNDICE E – Dashboard

```
        import streamlit as st
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        import pyodbc

        def season_select():
            cursor = conexao.cursor()
            comando_season = '''SELECT DISTINCT SEASON FROM shot_data'''
            cursor.execute(comando_season)
            resultado_season = cursor.fetchall()

            lista_season = []
            for item in resultado_season:
                for nome in item:
                    lista_season.append(nome)

            lista_season.sort()

            return lista_season

        def team_name():
            cursor = conexao.cursor()
            comando_team = f'''SELECT DISTINCT TEAM_NAME FROM shot_data WHERE
SEASON={season_choice}'''
            cursor.execute(comando_team)
            resultado_teams = cursor.fetchall()

            lista_teams = []
            for item in resultado_teams:
                for nome in item:
                    lista_teams.append(nome)

            lista_teams.sort()

            return lista_teams

        def player_name():
```

```
            cursor = conexao.cursor()
            comando_players = f'''SELECT DISTINCT PLAYER_NAME FROM shot_data WHERE
(SEASON={season_choice} AND TEAM_NAME='{team_choice}')'''
            cursor.execute(comando_players)
            resultado_players = cursor.fetchall()

            lista_players = []
            for item in resultado_players:
                for nome in item:
                    lista_players.append(nome)

            lista_players.sort()

            return lista_players

        def shot_dados(season, team, player):
            cursor = conexao.cursor()
            comando_dados = f'''SELECT LOC_X, LOC_Y, EVENT_TYPE FROM shot_data WHERE
(SEASON = {season} AND TEAM_NAME = '{team}' AND PLAYER_NAME = '{player}') '''
            cursor.execute(comando_dados)
            resultado_dados = cursor.fetchall()

            lista_dados = []
            for item in resultado_dados:
                lista_dados.append(item)

            dados = pd.DataFrame.from_records(lista_dados, columns=['LOC_X',
'LOC_Y', 'EVENT_TYPE'])

            return dados

        def plot_shot(dados):
            markers = {'Missed Shot': 'X', 'Made Shot': 'o'}
            paleta = {'Missed Shot': '#F0421D', 'Made Shot': '#36AC13'}
            fig, ax = plt.subplots(figsize=(4, 4))
            ax.imshow(im, extent=[-251, 250, -50, 860])
            sns.scatterplot(ax=ax, x=dados['LOC_X'], y=dados['LOC_Y'],
style=dados['EVENT_TYPE'], hue=dados['EVENT_TYPE'], s=15,
                            palette=paleta, markers=markers)
            lng = ax.legend(loc='upper left', title='Shots:', prop={'size': 6})
            lng._legend_box.align = 'left'
            plt.ylabel('')
            plt.xlabel('')
            plt.setp(ax.get_legend().get_texts(), fontsize='6')
            plt.setp(ax.get_legend().get_title(), fontsize='7')
            ax.set_yticklabels([])
            ax.set_xticklabels([])

            return fig

        def count_shot_att(season, team, player):
            cursor = conexao.cursor()
            comando_shot_att = f'''SELECT EVENT_TYPE FROM shot_data WHERE (SEASON =
{season} AND TEAM_NAME = '{team}' AND PLAYER_NAME = '{player}') '''
            cursor.execute(comando_shot_att)
            resultado_shot_att = cursor.fetchall()

            lista_shot_att = []
            for item in resultado_shot_att:
                for i in item:
                    lista_shot_att.append(i)

            return lista_shot_att

        def count_shot_made(season, team, player):
            cursor = conexao.cursor()
```

```
            comando_shot_made = f'''SELECT EVENT_TYPE FROM shot_data WHERE (SEASON =
{season} AND TEAM_NAME = '{team}' AND PLAYER_NAME = '{player}' AND EVENT_TYPE =
'Made Shot') '''
            cursor.execute(comando_shot_made)
            resultado_shot_made = cursor.fetchall()

            lista_shot_made = []
            for item in resultado_shot_made:
                for i in item:
                    lista_shot_made.append(i)

            return lista_shot_made

        def shot_made_area(season, team, player):
            cursor = conexao.cursor()
            comando_made_area = f'''SELECT EVENT_TYPE, ACTION_TYPE, SHOT_TYPE,
SHOT_ZONE_BASIC FROM shot_data WHERE (SEASON = {season} AND TEAM_NAME = '{team}'
AND PLAYER_NAME = '{player}' AND EVENT_TYPE = 'Made Shot') '''
            cursor.execute(comando_made_area)
            resultado_made_area = cursor.fetchall()

            lista_made_area = []
            for item in resultado_made_area:
                lista_made_area.append(item)

            dados_made_area = pd.DataFrame.from_records(lista_made_area,
columns=['EVENT_TYPE', 'ACTION_TYPE', 'SHOT_TYPE', 'SHOT_ZONE_BASIC'])

            return dados_made_area

        def shot_att_area(season, team, player):
            cursor = conexao.cursor()
            comando_att_area = f'''SELECT EVENT_TYPE, ACTION_TYPE, SHOT_TYPE,
SHOT_ZONE_BASIC FROM shot_data WHERE (SEASON = {season} AND TEAM_NAME = '{team}'
AND PLAYER_NAME = '{player}') '''
            cursor.execute(comando_att_area)
            resultado_att_area = cursor.fetchall()

            lista_att_area = []
            for item in resultado_att_area:
                lista_att_area.append(item)

            dados_att_area = pd.DataFrame.from_records(lista_att_area,
columns=['EVENT_TYPE', 'ACTION_TYPE', 'SHOT_TYPE', 'SHOT_ZONE_BASIC'])

            return dados_att_area

        def shot_miss_area(season, team, player):
            cursor = conexao.cursor()
            comando_miss_area = f'''SELECT EVENT_TYPE, ACTION_TYPE, SHOT_TYPE,
SHOT_ZONE_BASIC FROM shot_data WHERE (SEASON = {season} AND TEAM_NAME = '{team}'
AND PLAYER_NAME = '{player}' AND EVENT_TYPE = 'Missed Shot') '''
            cursor.execute(comando_miss_area)
            resultado_miss_area = cursor.fetchall()

            lista_miss_area = []
            for item in resultado_miss_area:
                lista_miss_area.append(item)

            dados_miss_area = pd.DataFrame.from_records(lista_miss_area,
columns=['EVENT_TYPE', 'ACTION_TYPE', 'SHOT_TYPE', 'SHOT_ZONE_BASIC'])

            return dados_miss_area

        def time_on_the_clock(season, team, player):
            cursor = conexao.cursor()
```

```
            comando_clock = f'''SELECT EVENT_TYPE FROM shot_data WHERE (SEASON =
{season} AND TEAM_NAME = '{team}' AND PLAYER_NAME = '{player}' AND EVENT_TYPE =
'Made Shot' AND PERIO = 4 AND MINUTES_REMAINING < 2) '''
            cursor.execute(comando_clock)
            resultado_clock = cursor.fetchall()

            lista_clock= []
            for item in resultado_clock:
                for i in item:
                    lista_clock.append(i)

            return lista_clock

        #Conectando ao Banco de Dados

        dados_conexao = (
            "Driver={SQL Server};"
            "Server=NOME_DO_SERVIDOR;"
            "Database=shot_dashboard;"
        )

        conexao = pyodbc.connect(dados_conexao)

        im = plt.imread("nba_full_court.jpg")
        sns.set()
        st.set_page_config(layout="wide")

        #Adicionado uma barra lateral ao dashboard, para o usuário escolher o que
deseja visualizar

        st.title('Interactive Dashboard')

        st.sidebar.markdown("### Filters")
        season_choice = st.sidebar.slider('Season:', min_value=min(season_select()),
max_value=max(season_select()), step=1)
        team_choice = st.sidebar.selectbox('Team:', team_name())
        player_choice = st.sidebar.selectbox('Player:', player_name())

        #st.markdown(f"### {player_choice}'s shotmap from the {season_choice}
season: ")
        st.markdown(f'### **{player_choice}**')
        st.markdown(f'{team_choice} - {season_choice}')
        col1, col2, col3, col4, col5 = st.columns([1,1,1,1,1])

        #Coluna 1, Shotmap

        col1.pyplot(plot_shot(shot_dados(season_choice, team_choice,
player_choice)))

        #Informações que serão mostradas na coluna 2
        df_shot_att_area = shot_att_area(season_choice, team_choice, player_choice)

        col2.metric(label='Shots Attempted', value=len(count_shot_att(season_choice,
team_choice, player_choice)))

        col2.metric(label='Shots made', value=len(count_shot_made(season_choice,
team_choice, player_choice)))

        percentage = len(count_shot_made(season_choice, team_choice,
player_choice))/len(count_shot_att(season_choice, team_choice, player_choice))*100
        col2.metric(label='Shot Percentage', value= f'{round(percentage, 2)}%')

        moda_shot_zone = df_shot_att_area['SHOT_ZONE_BASIC'].mode()
        col2.metric(label='Favorite zone', value= moda_shot_zone[0])

        best_shot_zone = df_shot_att_area.loc[df_shot_att_area['EVENT_TYPE'] ==
'Made Shot']['SHOT_ZONE_BASIC'].mode()
        col2.metric(label='Best zone', value= best_shot_zone[0])
```

```
        clutch_shots = len(time_on_the_clock(season_choice, team_choice,
player_choice))
        col2.metric(label='Clutch Shots', value=clutch_shots)


        #Informações que serão mostradas na coluna 3
        df_shot_miss_area = shot_miss_area(season_choice, team_choice,
player_choice)
        df_shot_made_area = shot_made_area(season_choice, team_choice,
player_choice)

        #Informações sobre arremessos para 2 pontos
        two_point_att =
len(df_shot_made_area.loc[df_shot_made_area['SHOT_TYPE']=='2PT Field Goal']) +
len(df_shot_miss_area.loc[df_shot_miss_area['SHOT_TYPE']=='2PT Field Goal'])
        col3.metric(label='2PT attempted', value=two_point_att)

        two_point_made =
len(df_shot_made_area.loc[df_shot_made_area['SHOT_TYPE']=='2PT Field Goal'])
        col3.metric(label='2PT Made', value=two_point_made)

        if two_point_att == 0:
            two_point_percentage = 0
        else:
            two_point_percentage =
len(df_shot_made_area.loc[df_shot_made_area['SHOT_TYPE']=='2PT Field
Goal'])/two_point_att*100
        col3.metric(label='2PT Percentage', value=f'{round(two_point_percentage,
2)}%')

        #Informações sobre arremessos para 3 pontos
        three_point_att =
len(df_shot_made_area.loc[df_shot_made_area['SHOT_TYPE']=='3PT Field Goal']) +
len(df_shot_miss_area.loc[df_shot_miss_area['SHOT_TYPE']=='3PT Field Goal'])
        col3.metric(label='3PT attempted', value=three_point_att)

        three_point_made =
len(df_shot_made_area.loc[df_shot_made_area['SHOT_TYPE']=='3PT Field Goal'])
        col3.metric(label='3PT Made', value=three_point_made)

        if three_point_att == 0:
           three_point_percentage = 0
        else:
            three_point_percentage =
len(df_shot_made_area.loc[df_shot_made_area['SHOT_TYPE']=='3PT Field
Goal'])/three_point_att*100
        col3.metric(label='3PT Percentage', value=f'{round(three_point_percentage,
2)}%')

        #Coluna 5 possui as logos do projeto, grupo e universidade, além do link
para o repositório
        col5.markdown(
                f'''
                <body style="background-color:Gray;">
                <p align="center">
                <img
src="https://github.com/crizmorais/shot_dashboard/blob/main/logo_cftb.jpeg?raw=true
" height="120" width="120"><br>
                <img
src="https://github.com/crizmorais/shot_dashboard/blob/main/logo_gefd.png?raw=true"
height="130" width="130"><br>
                <img
src="https://github.com/crizmorais/shot_dashboard/blob/main/logo_ufsc.png?raw=true"
height="150" width="125"><br><br>
                Developed by <br> Cristiano Zarbato Morais <br><br>
                <a href="https://github.com/crizmorais/shot_dashboard"> Github
Repository </a>
```

```
        </p></body>
''', unsafe_allow_html=True)
```