CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO ITAJAÍ - UNIDAVI

SAMUEL DOS SANTOS CHIODINI

PROTÓTIPO DE DASHBOARD COM A APLICAÇÃO DE DATA SCIENCE NO CAMPEONATO BRASILEIRO DE FUTEBOL: ANÁLISE E VISUALIZAÇÃO

RIO DO SUL
2023

CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO ITAJAÍ - UNIDAVI

SAMUEL DOS SANTOS CHIODINI

PROTÓTIPO DE DASHBOARD COM A APLICAÇÃO DE DATA SCIENCE NO CAMPEONATO BRASILEIRO DE FUTEBOL: ANÁLISE E VISUALIZAÇÃO

Trabalho de Conclusão de Curso a ser apresentado ao curso de Sistemas da Informação, da Área das Ciências Naturais, da Computação e das Engenharias, do Centro Universitário para o Desenvolvimento do Alto Vale do Itajaí, como condição parcial para a obtenção do grau de Bacharel em Sistemas de Informação.

Prof. Orientador: Dr. Marco Aurélio Butzke

RIO DO SUL
2023

CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO ITAJAÍ - UNIDAVI

SAMUEL DOS SANTOS CHIODINI

PROTÓTIPO DE DASHBOARD COM A APLICAÇÃO DE DATA SCIENCE NO CAMPEONATO BRASILEIRO DE FUTEBOL: ANÁLISE E VISUALIZAÇÃO

Trabalho de Conclusão de Curso a ser apresentado ao curso de Sistemas da Informação, da Área das Ciências Naturais, da Computação e das Engenharias, do Centro Universitário para o Desenvolvimento do Alto Vale do Itajaí - UNIDAVI, a ser apreciado pela Banca Examinadora, formada por:

_____

Professor Orientador: Dr. Marco Aurélio Butzke

Banca Examinadora:

_____

Prof.

_____

Prof.

Rio do Sul, 06 de mês de 2023.

Aos meus pais Conrado e Marilucia e a minha namorada Isadora que me apoiaram sempre.

AGRADECIMENTOS

Agradeço profundamente aos meus amigos e professores por todo o apoio que me ofereceram ao longo da minha jornada acadêmica. Quero expressar minha gratidão especial ao meu amigo David e à minha namorada Isadora, que foram fontes de incentivo quando precisei.

Agradeço ao professor e orientador deste trabalho extraordinário, Marco Aurélio Butzke, por ter gentilmente compartilhado seu tempo e conhecimento, desempenhando um papel fundamental na realização deste projeto de maneira excepcional. Sua orientação e colaboração foram inestimáveis e contribuíram significativamente para o sucesso deste trabalho.

Aos meus pais, quero dedicar um agradecimento especial por estarem sempre ao meu lado, apoiando minhas decisões e enfrentando comigo os momentos difíceis. Além disso, reconheço os inúmeros esforços que fizeram para me proporcionar a oportunidade de estudar e alcançar tudo o que conquistei. Pai e mãe, saibam que os amo profundamente e sou imensamente grato por tudo o que fizeram por mim. Obrigado por serem o alicerce da minha jornada.

RESUMO

O Campeonato Brasileiro de futebol, além de entreter, exerce impacto econômico, social e cultural. Este estudo propõe um protótipo de *dashboards* para analisar dados da tabela de classificação, utilizando técnicas de *Data Science*. O objetivo é criar uma ferramenta que proporcione uma classificação mais precisa e eficiente das equipes. A pesquisa é aplicada descritiva, seguindo a abordagem CRISP-DM em seis fases: compreensão do negócio, compreensão dos dados, preparação, modelagem, avaliação e desenvolvimento. O levantamento documental e de campo, com análise quali-quantitativa, contribuiu para responder como os *dashboards* podem aprimorar a classificação das equipes. Utilizando ferramentas como Microsoft Excel, *Visual Studio Code*, *Python*, *Pandas*, *NumPy*, Streamlit e *Google Forms*, a coleta e organização dos dados foram fundamentais. O *Visual Studio Code*, com Python, facilitaram o desenvolvimento enquanto o Streamlit criou interfaces amigáveis para a apresentação dos dados obtidos. Destaque para a importância da linguagem Python e bibliotecas como *Pandas* e *Numpy* que o tratamento dos dados e também o *Google Forms* que permitiu obter *feedback* dos usuários, contribuindo para a validação e futuras melhorias do protótipo. Este trabalho busca não apenas aprimorar a experiência de acompanhamento do campeonato, mas também promover acessibilidade aos dados, enriquecendo a experiência dos torcedores e gerando *insights* significativos. A aplicação de *Data Science* no esporte emerge com ferramenta valiosa para compreender e avaliar o desempenho das equipes, contribuindo para uma análise mais precisa e eficiente do Campeonato Brasileiro de futebol.

Palavras-Chave: *Dashboard, Data Science,* Campeonato brasileiro.

ABSTRACT

The Brazilian football championship, in addtion to providing entertainment, has economic, social and cultural impacts. This study proposes a prototype os dashboards to analyze data from the league table, employing Data Science techniques. The goal is to create a toll that offers a more accurate and efficient ranking of teams. The research is applied and descriptive, following the CRISP-DM approach in six phases: business understanding, data understanding preparation, modeling, evaluation and deployment. Documentary and field surveys, with a qualitative-quantitative analysis, contributed to answering how dashboards can enhance team rankings. Utilizing tools such as Microsoft Excel, Visual Studio Code, Python, Pandas, NumPy, Streamlit and Google Forms, data collection and organization were crucial. Visual Studio Code, with Python, facilitated development, while Streamlit created user-frindly interfaces for presenting the obtained data.The significance of the Python language and libraries like Pandas and NumPy for data processing, as well as Google Forms for obtaining user feedback, is higlighted. This contributed to the validation and future improvements of the prototype. This work aims not only to enhance the experience of following the championship but also to promote data acessibility,enriching the fans' experience and gerenating meaningful insights. The application of Data Sicence in sports emerges as a valuable tool for unsdertanding and evaluating team performance, contributing to a more precise and efficiente analusis of the Brazilian football championship.

Keywords: *Dashboard, Data Science, Brazilian chamopionship*.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

NumPy	*Numeric Python*
CRISP-DM	*Cross Industry Standart Process for Data Mining*
IA	Inteligência Artificial
ML	*Machine Learning*
VSCode	*Visual Studio Code*

SUMÁRIO

1. INTRODUÇÃO

O Campeonato Brasileiro de futebol é uma das competições mais prestigiadas e aguardadas pelos torcedores em todo o país. Além de proporcionar entretenimento e emoção, o campeonato possui impactos significativos nas esferas econômica, social e cultural das regiões onde os clubes estão sediados. Nesse contexto, surge a necessidade de criar ferramentas e serviços que possam auxiliar na análise e classificação das equipes, permitindo um acompanhamento mais detalhado e preciso dos resultados do campeonato.

Nesse sentido, a aplicação de Data Science tem se destacado como uma poderosa ferramenta para fornecer *insights* valiosos no campo esportivo. Através da análise de dados estatísticos, modelos preditivos e algoritmos de classificação é possível extrair informações relevantes que podem melhorar a compreensão e avaliação do desempenho das equipes durante o campeonato.

O presente trabalho de conclusão de curso tem como objetivo a aplicação de técnicas e métodos de *Data Science* na análise dos dados provenientes do Campeonato Brasileiro de Futebol. Além disso, busca-se a criação de representações visuais desses dados, visando fornecer uma avaliação precisa e atualizada do desempenho das equipes, possibilitando um acompanhamento mais detalhado e informativo.

Para alcançar esse resultado, será elaborado uma base de dados com os jogos e resultados destes confrontos, informações relevantes da partida com a aplicação de técnicas de *Data Science*, como análise estatísticas, *machine learning* e modelagem preditiva. Os resultados obtidos serão analisados, destacando sua contribuição para uma melhor compreensão do campeonato das equipes.

1.1 PROBLEMA DE PESQUISA

Como a utilização de *dashboards* baseados em rotinas de Data Science pode proporcionar uma classificação mais precisa e eficiente das equipes no Campeonato Brasileiro de Futebol?

1.2 OBJETIVOS

1.2.1 Geral

- Desenvolver um protótipo com dashboards para análise dos dados da tabela de classificação do campeonato nacional de futebol, utilizando técnicas de *Data Science*.

1.2.2 Específicos

- Levantar os resultados das partidas do campeonato brasileiro de futebol para elaborar uma tabela básica de classificação.
- Verificar técnicas de *Data Science* que podem ser aplicadas para construção dos *dashboards*.
- Implementar as fases de enriquecimento dos dados para cada técnica de *Data Science* selecionada para o modelo de inteligência.
- Implementar as rotinas do modelo de inteligência e os *dashboards* para visualização dos resultados.
- Validar o modelo de inteligência e os *dashboards* por meio de entrevista.

1.3 JUSTIFICATIVA

O campeonato nacional de futebol é uma competição importante não apenas para os times envolvidos, mas também para a sociedade em geral proporcionando entretenimento e muita emoção para os torcedores. A competição traz benefícios econômicos, sociais e culturais para as regiões onde os clubes são pertencentes. Além disso, o futebol nacional promove a união e a confraternização entre as diferentes regiões do país gerando um clima de euforia e entusiasmo que se estende por todo o país.

A torcida é um elemento fundamental no futebol, acompanhando seus times de perto em todos os jogos, porém é comum que estes não apenas acompanham somente seu time favorito e sim o campeonato como um todo, verificando os placares dos demais jogos e outras informações relevantes ao campeonato.

Com o intuito de aprimorar a experiência de acompanhamento do campeonato, o protótipo proposto tem como finalidade fornecer informações detalhadas sobre a competição,

bem como sobre os times individualmente. Isso será realizado por meio de rotinas que disponibilizarão *dashboards* relevantes durante as rodadas.

O projeto visa atender à acessibilidade aos dados, facilitando o acompanhamento dos jogos e proporcionando *insights* valiosos aos torcedores, promovendo assim uma experiência mais rica em detalhes e envolvente para todos os amantes do futebol nacional.

2. REFERENCIAL TEÓRICO

Neste capítulo serão abordadas as definições e opiniões de diferentes autores referentes às tecnologias utilizadas para o desenvolvimento da pesquisa, entre elas: *Python*, *Streamlit*, *Numpy*, *Pandas* e outras.

2.1 DESENVOLVIMENTO DE SOFTWARE

De acordo com Valente (2020), desenvolvimento de software se refere à implementação do sistema, onde diversas decisões precisam ser tomadas, como a definição de algoritmos e estruturas de dados a serem usados, frameworks e bibliotecas de terceiros, técnicas para tratamento de exceções, padrões de nomes, layout e documentação de código, e ferramentas a serem utilizadas no desenvolvimento, como compiladores, ambientes integrados de desenvolvimento, depuradores, sistemas gerenciadores de bancos de dados e ferramentas para construção de interface.

2.2 LINGUAGEM DE PROGRAMAÇÃO

Para Haverbeke (2018), a programação é uma ação de criar programas, que consiste em um conjunto de instruções precisas.

Uma linguagem de programação é uma linguagem artificial criada para computadores. Com a combinação de palavras e frases para expressar conceitos e instruir o computador a executar determinadas tarefas.

Como os computadores não possuem inteligência própria, é necessário ser preciso e detalhado em cada instrução que é dada ao computador, tornando a programação um processo complexo.

2.3 PYTHON

Segundo Cruz (2015), *Python* é uma linguagem de programação de alto nível e interpretada, que suporta vários paradigmas de programação, incluindo imperativo, orientado a objetos e funcional. Ela tem tipagem dinâmica e forte, escopo léxico e gerenciamento automático de memória. Além disso, *Python* oferece várias estruturas de dados embutidas na sintaxe, como tuplas, listas e dicionários, o que torna o código mais expressivo. A linguagem

também possui uma biblioteca padrão muito ampla e poderosa, conhecida como "baterias inclusas".

O autor afirma que o *Python* tem uma sintaxe simples que pode ser facilmente aprendida, e com prática, elementos mais complexos, como *comprehensions*, *lambdas*, *packing* e *unpacking* de argumentos. Uma característica incomum da linguagem é o uso da indentação para definir blocos de código. A comunidade Python valoriza muito a legibilidade do código e tem dois elementos que reforçam esse princípio: PEP-8 e *The Zen of Python*.

> Apesar de ser uma linguagem interpretada, existe um processo de compilação transparente que transforma o código texto em *bytecode*, que, por sua vez, é interpretado por uma virtual *machine* (VM). A implementação padrão da linguagem Python é chamada de *CPython* e, apesar de existirem outras implementações da especificação. (CRUZ, 2015, p. 18).

O *Python* é uma linguagem que é utilizada em diversos contextos, incluindo o desenvolvimento de aplicativos web, análise de dados, aprendizado de máquina e automação de tarefas (Python.org).

O uso da linguagem no desenvolvimento web é preciso adicionar frameworks com como o Django e Flask que são os mais populares na área fornecendo recursos e ferramentas para desenvolvedores web, permitindo a criação de aplicativos escaláveis e seguros (Real Python).

O *Python* é uma das linguagens mais usadas em análise de dados e ciência de dados em conjunto de bibliotecas populares como o *Numpy*, *Pandas*, *Matplotlib* e *SciPy*. Com essas bibliotecas os cientistas de dados, podem limpar, visualizar grandes conjuntos de dados e aplicar técnicas avançadas de aprendizado de máquina e inteligência artificial.

Na área de automação de tarefas o *Python* é frequentemente usado para automatização de tarefas repetitivas, principalmente as que possuem relação com à administração de sistemas e redes, mas para isso é necessário utilizar a biblioteca Ansible que auxilia na configuração e gerenciamento de servidores e infraestrutura de redes (Real Python).

Para o aprendizado de máquina e inteligência artificial o *Python* se tornou uma das linguagens mais populares em conjunto de bibliotecas famosas como TensorFlow, Keras, PyTorch e ScikitLearn. Estas permitem que desenvolvedores criem e treinem modelos para o aprendizado de máquina e implantem soluções avançadas de inteligência artificial (Analytics Vidhya).

O site Itecnologia (2021), disponibiliza um conteúdo onde é possível verificar as vantagens de desvantagens da utilização da linguagem *Python*. No Quadro 1 estão disponíveis algumas vantagens e desvantagens da linguagem *Python*.

Quadro 1- Vantagens e Desvantagens na utilização do *Python*

Vantagens	**Desvantagens**
Como o *Python* é uma linguagem de programação de alto nível, se torna uma linguagem fácil de ler e entender.	O *Python* é ruim para mobile, pois tem um baixo poder de processamento se comparado com outras linguagens para dispositivos móveis.
Uma linguagem interpretada significa que o código é executado linha por linha, em caso de algum erro ele para sua execução e avisa qual erro que ocorreu, e o *Python* funciona dessa forma.	No *Python* o tipo de variável pode ser modificado a qualquer momento, isso pode ser uma vantagem, como também uma desvantagem, pois essa situação se não for bem controlada pode causar Runtime Errors.
O *Python* é uma linguagem que pode ser utilizada para diversas plataformas como Windows, Linux, Mac, Android… Isso é possível por ela ser uma linguagem interpretada, ou seja, outras máquinas virtuais podem interpretar seu código.	Para prover toda sua simplicidade para o programador, a linguagem de Programação *Python* utiliza muita memória e um péssimo gerenciamento de memória. Isso é uma desvantagem séria quando se vai desenvolver aplicações que requerem uma boa otimização de memória.

Fonte: Quadro desenvolvido a partir do site Itecnologia (2021).

2.3.1 PEP-8

Segundo a documentação do PEP8 (2001), é um guia de estilos de código *Python* que é amplamente empregado, define as convenções e recomendações para tornar o código mais legível e consistente. Tem como seu principal objeto tornar os códigos feitos em *Python* mais fáceis de entender e algumas de suas recomendações é a utilização de espaços em branco, endentação, nomenclatura de variáveis, funções e módulos e a limitação no comprimento das linhas de código.

2.3.2 The Zen of Python

O *The Zen of Python* é um pequeno texto que fala muito sobre o estilo de programação em *Python*:

> Bonito é melhor que feio. Explícito é melhor que implícito. Simples é melhor que complexo. Complexo é melhor que complicado. Linear é melhor que aninhado. Esparso é melhor que denso. Legibilidade conta. Casos especiais não são especiais o suficiente para quebrar regras. Embora praticidade prevaleça sobre pureza. Erros nunca devem ser silenciados. A não ser explicitamente. Diante de uma ambiguidade, não caia na armadilha do chute. Deve existir um – e preferencialmente um – jeito óbvio de se fazer algo. Embora possa não parecer óbvio a não ser que você seja holandês. Agora é melhor que nunca. Embora nunca normalmente seja melhor que exatamente agora. Se a implementação é difícil de explicar, ela é uma má ideia. Se a implementação é fácil de explicar, talvez seja uma boa ideia. Namespaces são uma grande ideia – vamos usá-los mais! (CRUZ, 2015, p. 19).

De acordo com PETERS (2004), criador do texto e da linguagem *Python* o *Zen Of Python* consiste em 19 aforismos que abrangem questões como simplicidade, legibilidade e praticidade. Criado para auxiliar os desenvolvedores a escreverem códigos claros e concisos sem a existência de complexidade desnecessária.

2.4 PYTHON PARA DATA SCIENCE

De acordo com VanderPlas (2016), o *Python* para Data Science não é uma nova linguagem de programação, mas sim uma abordagem específica para usar a linguagem em análise de dados.

A linguagem adaptada para Data Science é formada pela junção de bibliotecas específicas utilizadas para a análise de dados, machine learning e visualização de dados. Com essas bibliotecas é possível realizar tarefas complexas de análise de dados de uma forma mais rápida e fácil.

2.4.1 Streamlit

Streamlit (2020), se define como uma biblioteca de código aberto para *Python* que tem como objetivo tornar a criação de aplicativos da web acessível para todos, especialmente para cientistas de dados e engenheiros de aprendizado de máquina, compartilhando páginas web customizadas.

A biblioteca é considera fácil de usar, com uma sintaxe simples e intuitiva, permitindo que desenvolvedores *Python* possam desfrutar de criações de interfaces gráficas interativas, filtragem e ordenação de dados, além de suporte para widgets de entrada, como botões, caixas de seleção e campos de entrada de texto.

> Para construir um aplicativo da web simples no Streamlit, tudo o que um desenvolvedor precisa fazer é importar a biblioteca, criar uma função para renderizar a interface do usuário e, em seguida, executar o aplicativo com um único comando. Um exemplo disso é criar um aplicativo que permite ao usuário selecionar um arquivo CSV e exibir seu conteúdo em uma tabela interativa. Em poucas linhas de código, um desenvolvedor pode criar uma interface do usuário para seleção de arquivos, carregar o arquivo selecionado e exibir os dados em uma tabela interativa. AWARI (2023, n.p.).

2.4.2 NumPy

De acordo com VanderPlas (2016), a biblioteca NumPy foi criado em 2005 por Travis oliphant e tem o código aberto para utilizar com a linguagem *Python*, seu papel principal é fornecer suporte para matrizes e operações matemáticas de alta performance. O NumPy permite lidar com grandes quantidades de dados, como por exemplo, em aplicações de ciência de dados ou aprendizado de máquina. Além disso, a biblioteca algumas funções para a manipulação desde dados, como ordenação, filtragem e transformação, bem como funções para estatísticas e probabilidade.

> Além dos recursos mais voltados a aplicações de análise de dados, no NumPy Python você também encontrará funções matemáticas para operações rápidas em arrays, sem a necessidade de escrever laços, recursos de álgebra linear, geração de números aleatórios, transformadas de Fourier, ferramentas para trabalhar com dados mapeados em memória, como também uma API para conectar o NumPy Python a bibliotecas escritas em C, C++ e FORTRAN. (HARVE, 2023, n.p.).

"O *NumPy* é frequentemente usado em conjunto com outras bibliotecas de ciência de dados em *Python*, como *Pandas*, *SciPy* e *Matplotlib*, para análise de dados, visualização e modelagem estatística." (VANDERPLAS, 2016 p.548).

2.4.3 Pandas

Pandas (2021) é uma biblioteca de código aberto para *Python* usada na análise de dados em tabelas, fornecendo duas estruturas de dados principais, a série e o *Dataframe*, com estas é possível a manipulação de dados em tabelas bidimensionais.

Com a biblioteca possui várias funções de leitura e escrita de dados em vários formatos, CSV, Excel e SQL. Contendo também ferramentas para limpeza e transformação de dados, como preenchimento de valores faltantes, remoção de dados duplicados e até a filtragem de dados.

> "O Pandas também inclui ferramentas para agregação de dados, como agregação de dados por grupo, e operações de *join* e merge. A biblioteca permite a visualização de dados com gráficos, e possui integração com outras bibliotecas populares de ciência de dados em Python, como *NumPy* e *Matplotlib*.". (PANDAS, 2021, n.p.).

No site da Alura (2023), disponibilizaram o material para realizar a comparação entre as estruturas principais do *Pandas*, series e *Dataframe*. No Quadro 2 estão disponíveis as comparações.

Quadro 2 - Diferenças entre *Series* e *Dataframe*

Series	**DataFrame**
As *Series* são objetos de tipo *array* unidimensional, com um eixo de rótulos, também chamado de index, que é responsável por identificar cada registro. Um exemplo de Series no Pandas é encontrado no *dataset* Iris quando isolamos uma das variáveis para exibição	Os *DataFrames* são objetos bidimensionais, de tamanho variável. O seu formato é de uma tabela, onde os dados são organizados em linhas e colunas. Além disso, enquanto podemos pensar a Series como uma única coluna, o *DataFrame* seria uma união de várias Series sob um mesmo index.

Fonte: Quadro desenvolvido a partir do site da Alura (2023).

2.4.4 Altair

Altair (2019) se define como uma biblioteca de código aberto e de visualização declarativa para Python, baseada em Veja e Veja-Lite. Ele oferece uma gramática poderosa e concisa que permite construir rapidamente uma ampla gama de visualizações estáticas. A API do Altair é simples, amigável, consistente e construída sobre a poderosa gramática de visualização Veja-Lite.

2.4.5 Scikit-Learn

De acordo com Jolly (2018) fala que o Scikit-Learn é um software gratuito com o código aberto que que ajuda a resolver problemas com seus algoritmos de aprendizado de máquina supervisionados e não supervisionados.

O autor também esclarece que a razão primordial para a imensa popularidade do Scikit-Learn reside no fato de que a maioria dos algoritmos de aprendizado de máquina mais amplamente adotados no mundo podem ser facilmente implementados por meio de uma abordagem de '*plug-and-play*'.

2.5 DATA SCIENCE

Segundo VanderPlas (2016), *Data Science* é uma área de estudo interdisciplinar com a combinação de várias técnicas de estatísticas, matemática, programação e visualização de dados para extrair conhecimento de grande conjunto de dados, onde serão utilizados para auxiliar na tomada de decisão em diversas áreas onde a ciência de dados é aplicada.

O autor afirma que para uma boa aplicação de ciência de dados é necessário aplicar diversas técnicas de estatísticas e computacionais, como por exemplo, aprendizado de máquina e mineração de dados, com isso é possível descobrir padrões, realizar previsões e modelar comportamentos dos dados. Além disso, a ciência de dados se concentra em todos os aspectos de ciclo de vida dos dados, desde a coleta até a análise e comunicação dos resultados.

> "É importante destacar que, além da complexidade técnica, a ciência de dados tem um importante papel em um contexto de negócios. Conforme destacam Provost e Fawcett (2013), é essencial compreender as perguntas de negócios que precisam ser respondidas e os objetivos que se deseja atingir antes de começar a trabalhar com os dados. A análise de dados deve estar alinhada com a estratégia de negócios e ser capaz de gerar insights relevantes para tomadas de decisão.". (FARIAS, 2020, n.p.).

Conforme apontado pelos autores Provost e Fawcett (2013), a Data Science é uma ferramenta extremamente poderosa para a obtenção de previsões futuras, pois permite a análise de grandes volumes de dados em tempo real e o desenvolvimento de modelos baseados em algoritmos de aprendizado de máquina. Os autores destacam que a aplicação de técnicas de Data Science para previsões futuras pode ser útil em diversas áreas, como marketing, finanças, saúde e muitas outras.

2.5.1 Machine Learning

Para Alpaydin (2010), *Machine Learning* é a área de estudo que se dedica a desenvolver e transformar técnicas e algoritmos para que computadores possam aprender a partir de dados e experiências passadas e utilizar esse conhecimento para tomar decisões e realizar tarefas.

De acordo com o autor essas técnicas incluem algoritmos de aprendizagem supervisionada, não supervisionada e por reforço, além de técnicas de pré-processamento e seleção de características dos dados.

Goodfellow, Bengio e Courville (2016), destaca que o *Machine Learning* é uma das áreas que está em constante evolução, obtendo novas técnicas e modelos regularmente. Para os

autores, com a diversidade de técnicas e modelos disponíveis para utilizar se torna um dos principais desafios da área, pois é necessário saber selecionar a técnica correta para cada desenvolvimento.

Segundo um relatório publicados pela consultoria International Data Corporation (2019), é esperado que para os próximos anos o mercado de *Machine Learning* cresça depressa. A receita global com os produtos e serviços com *Machine Learning* deve crescer a uma taxa anual. A tecnologia é fundamental para a análise de grandes volumes de dados sendo utilizada cada vez mais em diversas áreas.

2.6 ESTATÍSTICA

Segundo Triola (2010), é uma área da matemática que utiliza métodos qualitativos para coletar, analisar e interpretar dados com o principal objetivo de fornecer algumas técnicas para auxiliar na tomada de decisão baseada em informações numéricas, buscando identificar padrões, tendências e relacionamentos. Utilizada em diversas áreas como em medicina, engenharia, negócios, ciência de dados e outras.

De acordo com Costa Neto (2018), destaca que a Estatística pode ser dividida em duas áreas principais: a Estatística Descritiva, que se preocupa com a organização e descrição dos dados experimentais, e a Estatística Indutiva, que se concentra na análise e interpretação desses dados. Com essas duas abordagens, a Estatística pode ser uma ferramenta valiosa para profissionais de diversas áreas, permitindo que sejam feitas análises mais precisas e tomadas decisões mais embasadas.

2.6.1 Estatística descritiva

O site FM2S (2017), indica que podemos utilizar a estatística descritiva para comparar grupos como por exemplo a média de idade de homens e mulheres em uma população, visualizar com a criação de histograma para mostrar a distribuição de uma variável em um conjunto de dados, identificar padrões como os outliers ou valores atípicos em um conjunto de dados.

2.7 CRISP-DM

Para os autores Goldschmidt e Passos (2015), o modelo de mineração de dados CRISP-DM tem como seu principal objetivo a obtenção de dados. Ele fornece uma estrutura de organização sequencial para orientar o desenvolvimento de projetos, separando-o em seis fases, conforme evidenciado no Quadro 3.

Quadro 3 - Fases do CRISP-DM

Fase	**Definição**
Compreensão do negócio	A primeira fase é entender os objetivos do negócio, as necessidades dos stakeholders e as metas do projeto de mineração. Envolve a definição clara do problema a ser resolvido, fazer um levantamento do hardware e do software existentes, inventário das bases de dados disponíveis, verificar a existência de Data Warehouses, identificar e documentar todo conhecimento prévio.
Compreensão dos dados	Para essa fase o foco é explorar e familiarizar-se com os dados disponíveis para o projeto. Coletar informações sobre a estrutura dos dados, sua qualidade, distribuição e demais características.
Preparação dos dados	Os dados são preparados para a análise, envolvendo a seleção de dados, limpeza, integração, transformação e a construção do conjunto de dados para o modelo.
Modelagem	Consiste na escolha e aplicação técnica de modelagem sobre os dados a serem analisados. São selecionados os algoritmos de mineração de dados adequados para a construção do modelo
Avaliação	Nessa etapa é feita a construção e avaliação do modelo sobre a qualidade e eficácia em resolver o problema. Aqui são aplicadas métricas de desempenho e a realização de testes para garantir que o modelo atendo os critérios estabelecidos.
Desenvolvimento	Para última etapa são evidenciados os resultados e possíveis alternativas de ação, descobrimento e conhecimento na organização.

Fonte: Quadro desenvolvido a partir de Goldschmidt e Passos (2015).

É importante ressaltar que as fases apresentadas no Quadro 3 são de extrema importância para a aplicação do modelo CRISP-DM na busca por conhecimento dos dados. Devido à natureza iterativa do modelo, é comum retornar às etapas anteriores durante o desenvolvimento do projeto à medida que novos dados são descobertos. Quando aplicado corretamente, o modelo tem o potencial de melhorar significativamente a eficiência e eficácia na geração de conhecimento.

2.8 METODOLOGIAS DE INTELIGÊNCIA

Para o autor Mitchell (1997), os métodos de inteligência referem-se a abordagens sistemáticas e estruturadas para o desenvolver e implementar soluções baseadas em inteligência artificial (IA). Essas metodologias envolvem a aplicação de técnicas e algoritmos de IA para

resolver problemas complexos, extrair informações úteis dos dados e tomar decisões informadas.

2.8.1 Clusterização

De acordo com os autores Han, Kamber e Pei (2011), a clusterização conhecida também como a análise de agrupamento é uma técnica de aprendizado não supervisionado em mineração de dados. Ela envolve a organização de um conjunto de objetos em grupos ou clusteres, onde objetos semelhantes são agrupados juntos com base em suas características compartilhadas. O objetivo da clusterização é identificar padrões nos dados, descobrindo estruturas ocultas e relacionamentos entre os objetos, mesmo sem ter rótulos ou categorias predefinidas.

Os altores escrevem que os algoritmos de clusterização são aplicados em uma variedade de domínios, como segmentação de clientes, classificação de documentos, análise de imagens genômica, entre outras. Essa técnica permite a descoberta de grupos naturais e distintos nos dados, facilitando a compreensão, organização e análise posterior dos mesmos.

Ressaltam sobre a importância da clusterização na análise de dados, pois com esses métodos é possível organizar grandes volumes de dados em grupos ou clusteres, proporcionando uma compreensão mais profunda dos padrões e estruturas presentes nos conjuntos de dados.

2.8.2 Regressão Linear

Os autores James, Hastie e Tibshirani (2013), afirmam que a regressão linear é uma técnica fundamental na área de Data Science, pois é muito utilizada para análise preditiva e modelagem de dados, uma abordagem estatística que busca estabelecer uma relação linear entre uma variável dependente ou independente. Na área de Data Science a regressão linear é aplicada em várias etapas o processo de análise e dados, comumente utilizada na fase de pré-processamento.

Destacam ainda que a regressão linear também é utilizada para avaliar a importância das variáveis independentes, por meio da análise dos coeficientes e regressão. Com isso é possível verificar a magnitude do impacto das variáveis independentes na variável dependente e assim identificas quais são as variáveis mais relevantes para o modelo.

Segundo os autores a combinação da regressão linear juntos com outras técnicas de algoritmos se tornam essências para o Data Science, fornecendo uma abordagem estatística para modelagem preditiva e análise de dados. A regressão permite entender as relações entre as variáveis, prever futuros e identificar a importância das variáveis independentes no contexto de um problema específico.

No site Data Science (2020), fornece informações para o entendimento entre a diferença de variáveis independentes e dependentes. No Quadro 4 é possível verificar essas diferenças.

Quadro 4 - Diferença entre variáveis independentes e dependentes

Variável independente	Variável dependente
Uma variável independente é a variável que é alterada ou controlada em um experimento científico para testar os efeitos sobre a variável dependente.	Uma variável dependente é a variável que está sendo testada e medida em um experimento científico.

Fonte: Quadro desenvolvido a partir do site Data Science (2020).

"A variável independente é aquela que o experimentador controla. A variável dependente é a variável que se ajusta por causa da variável independente." (DATA SCIENCE, 2020, n.p.).

"As duas variáveis podem estar conectadas por circunstâncias e resultados lógicos. Se a variável independente muda e elas estão relacionadas, então a variável dependente é influenciada." (DATA SCIENCE, 2020, n.p.).

2.8.3 Regressão Logística

No site da AWS (2023), diz que a regressão logística é uma técnica de análise que usa a matemática para encontrar as relações entre dois fatores de dados. Em seguida, essa relação é empregada para prever o valor de um desses fatores com base no outros. Geralmente, as previsões resultam em número limitado de categorias, como "sim" ou "não".

Ainda no site informa que a regressão logística desempenha um papel fundamental no domínio da inteligência artificial e do aprendizado de máquina (IA/ML). Os modelos de IA/ML constituem programas de software que podem ser treinados para executar tarefas complexas de processamento de dados sem a necessidade de intervenção humana. Os modelos de IA/ML baseados em regressão logística auxiliam as organizações na extração de informações práticas a partir de seus dados comerciais. Essas informações podem ser empregadas na análise

preditiva, com o objetivo de reduzir custos operacionais, melhorar a eficiência e acelerar o crescimento de maneira significativa.

2.9 DASHBOARD

De acordo com Staggemeier (2023), *dashboard* é um painel que exibe os dados direto ao ponto, com visualizações rápidas e certeiras na identificação da informação para tomadas de decisões. São constituídos de gráficos analíticos, possuem recursos de interatividade, indicadores e métricas como apoio a tomada de decisão.

O autor diz que conceitualmente existem três modelos de *dashboards,* como indica o Quadro 5.

Quadro 5 - Modelos de *dashboards*

Model	**Conceito**
Operacionais	Usados para monitorias níveis.
Táticas	Usada para acompanhamento das operações que foram criadas com base nas estratégias.
Estratégicos	Usadas com indicadores ou KPI's Juntos das métricas e com um Scorecard é possível mensurar os resultados obtidos.

Fonte: Quadro desenvolvido a partir de STAGGEMEIER (2023).

2.10 NGROK

Segundo a documentação do Ngrok (2023), define que o software é um proxy reverso que é distribuído globalmente que protege e disponibiliza uma porta de entrada para o seu aplicativo local, podendo utilizá-lo para testes e conectividades seguras com nas redes cliente ou em dispositivos locais.

3. METODOLOGIA DA PESQUISA

A metodologia de pesquisa utilizada para o presente trabalho de conclusão de curso caracteriza-se como pesquisa aplicada e descritiva, pois o seu principal objetivo é desenvolver um protótipo de d*ashboards* com uso de *Data Science* para o campeonato brasileiro de futebol. O trabalho buscou responder os seguintes problemas: Como a utilização de *dashboards* baseados em rotinas de Data Science pode proporcionar uma classificação mais precisa e eficiente das equipes no Campeonato Brasileiro de Futebol?

Aos procedimentos utilizados, utilizou-se de levantamento documental e de campo com análise quali-quantitativa, em vista que foram analisadas as informações do campeonato brasileiro de futebol e dos times para a geração do protótipo de dashboards

O projeto se baseia no uso do modelo CRISP-DM, uma abordagem que orienta o processo de obtenção de conhecimento a partir de dados. Esse modelo é composto por seis fases principais com diz Goldschmidt e Passos (2015): compreensão do negócio, compreensão dos dados, preparação dos dados, modelagem, avaliação e desenvolvimento.

Durante a etapa de coleta de dados do campeonato, diversas ferramentas essenciais para o sucesso da pesquisa. O Microsoft Excel desempenhou um papel central na organização e estruturação dos dados coletados, permitindo uma abordagem eficiente na manipulação das informações relacionadas ao campeonato. Além disso, o site da cfb.com.br como fonte primária para obter resultados de jogos atualizados e confiáveis, sendo esses dados fundamentais para a pesquisa.

No desenvolvimento e tratamento dos dados, foi utilizado *Integrated Development Environment* (IDE) *Visual Studio Code*, juntamente com a linguagem de programação Python. As bibliotecas Pandas e Numpy também foram empregadas para análise e manipulação eficaz dos dados coletados. A parte visual do projeto foi desenvolvida com a utilização da biblioteca Streamlit, possibilitando a criação de interfaces amigáveis para apresentação dos dados de forma acessível e interativa.

Para a obtenção de *insights* e *feedback* dos usuários, foi utilizado o Google Forms, que permitiu coletar opiniões e sugestões relevantes para a melhoria do protótipo. Essa ferramenta desempenhou um papel importante na interação com os usuários e na validação do projeto.

O desenvolvimento deste trabalho se concentra principalmente na análise e utilização dos dados dos jogos do campeonato brasileiro de futebol. A obtenção, compreensão, preparação, modelagem, avaliação e uso desses dados são etapas fundamentais no processo de desenvolvimento da aplicação.

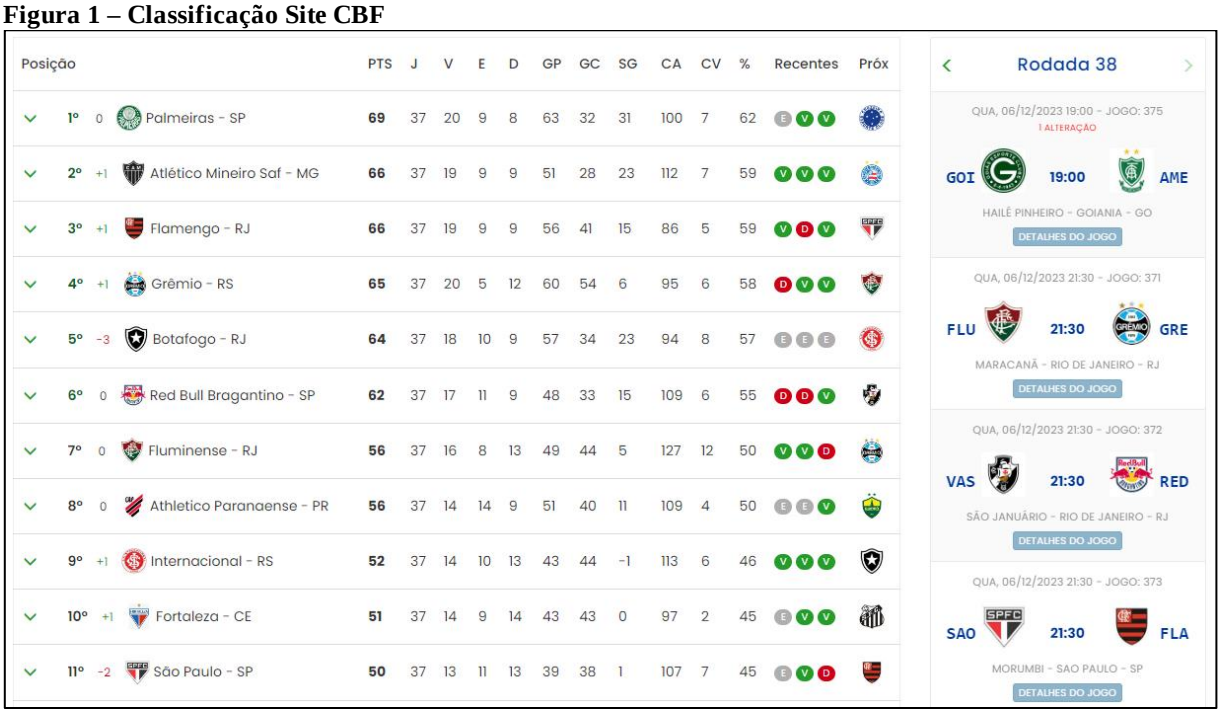
3.1 ESTADO DA ARTE

Nesta seção é descrito um site já existente nos mercados que se assemelha à proposta do desenvolvimento deste projeto.

3.1.1 CBF – Campeonato Brasileiro de Futebol - Série A – 2023

O website da CBF disponibiliza uma seção dedicada ao Campeonato Brasileiro de Futebol na Série A, contendo informações sobre os jogos, resultados, a tabela de classificação e o desempenho das equipes em termos de vitórias e derrotas. Essa estrutura guarda semelhança com o protótipo proposto neste projeto. Veja na Figura 1.

Figura 1 – Classificação Site CBF



Fonte: CBF (2023).

Existem outros enumero sites que apresentam informações sobre o campeonato, jogos e time, porém o que se assemelha melhor com o protótipo proposto no projeto seria o do website da CBF.

4. APLICAÇÃO DA CIÊNCIA DE DADOS NO CAMPEONATO BRASILEIRO DE FUTEBOL: ANÁLISE E VISUALIZAÇÃO

Neste capítulo serão abordadas as principais etapas relacionadas a aplicação durante o desenvolvimento desse trabalho.

4.1 VISÃO GERAL DO SISTEMA

Este projeto focaliza a criação de uma plataforma de análise e visualização de dados que utiliza uma base de dados simples para gerar um dashboard abrangente sobre o Campeonato Brasileiro de Futebol. A abordagem visa fornecer uma visão geral do campeonato, enquanto permite análises detalhadas das equipes participantes. A base de dados, coletada de fontes confiáveis, é processada e transformada em um painel dinâmico que oferece informações essenciais sobre as rodadas, partidas, resultados e desempenho das equipes.

Além disso, o dashboard possibilita a análise individual de cada time, incluindo estatísticas, tendências e comparações de desempenho. Com essa ferramenta, os usuários têm acesso a uma visão abrangente e acessível do campeonato, ao mesmo tempo em que podem mergulhar em análises detalhadas para entender melhor o desempenho de suas equipes favoritas. Este trabalho representa uma valiosa fonte de informações para entusiastas do futebol, analistas esportivos e tomadores de decisões, fornecendo uma maneira intuitiva e eficaz de acompanhar e compreender o Campeonato Brasileiro de Futebol.

4.2 DEFINIÇÃO DE MODELOS E PROCESSOS

Para o desenvolvimento e criação da base de dados, foram definidos o uso de dois principais softwares: o VSCode para a criação do código fonte e o Excel para a criação da base de dados.

Para o desenvolvimento dos dashboards, é necessário estabelecer processos bem definidos, englobando a coleta de dados, análise dos mesmos, avaliação e validação dos resultados. A metodologia escolhida para conduzir todas essas etapas é o modelo de processo CRISP-DM, como apresentado no tópico de revisão da literatura.

4.3 COLETA DOS DADOS DO CAMPEONATO BRASILEIRO DE FUTEBOL

A seguir será apresentado a forma utilizada para obter os dados das disputas do campeonato.

4.3.1 Web Scraping

A coleta de dados foi realizada manualmente e conduzida utilizando uma fonte pública, o site cbf.com.br, que disponibiliza e exibe informações sobre as rodadas do Campeonato Brasileiro de Futebol. O foco principal da coleta de dados foi a obtenção das partidas e seus resultados correspondentes. A maneira que os jogos são apresentados pelo site podem ser observados na Figura 2.

Figura 2 - Apresentação dos jogos

Fonte: CBF (2023).

Todos os jogos da temporada de 2023 foram registrados totalizando 38 rodadas com 10 jogos por rodada gerando 380 confrontos. À medida que as rodadas se desenrolavam, os dados correspondentes eram inseridos em uma planilha do Excel. Nessa planilha, os registros eram categorizados em colunas distintas, abrangendo informações como a temporada, a rodada, o

time mandante, o resultado do mandante, o time visitante e o resultado do visitante. A maneira que os dados foram registrados na planilha é possível ser observada na Figura 3.

Figura 3 - Dados registrados na planilha

	A	B	C	D	E	F
1	**Temporada**	**Rodada**	**Mandante**	**Score_m**	**Visitante**	**Score_v**
2	2023	1	PAL	2	CUI	1
3	2023	1	AME	0	FLU	3
4	2023	1	CAP	2	GOI	0
5	2023	1	BOT	2	SAO	1
6	2023	1	RED	2	BAH	1
7	2023	1	FOR	1	INT	1
8	2023	1	CAM	1	VAS	2
9	2023	1	COR	2	CRU	1
10	2023	1	FLA	3	CFC	0
11	2023	1	GRE	1	SAN	0

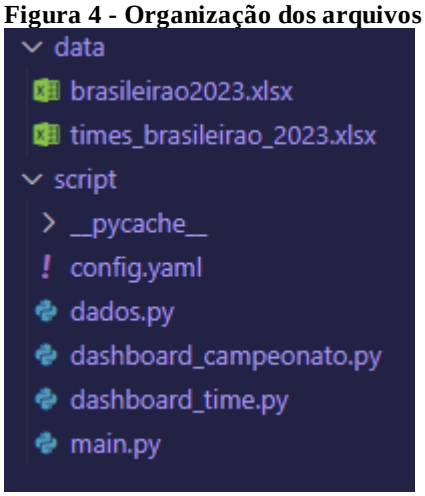
Fonte: O autor (2023).

Para obter esses dados da planilha, foi efetuado o download do arquivo no formato .xlsx, possibilitando, assim, a sua utilização no desenvolvimento de dashboards.

4.4 ORGANIZAÇÃO DOS ARQUIVOS

Para manter a organização, os arquivos foram distribuídos em pastas distintas. O arquivo de geração de dados da aplicação e a criação dos dashboards foram adicionados à pasta 'script', enquanto os arquivos relacionados aos jogos foram adicionados à pasta 'data', conforme ilustrado na Figura 4.

Figura 4 - Organização dos arquivos



Fonte: O autor (2023).

4.5 TRATAMENTO DOS DADOS

A etapa de tratamento de dados desempenha um papel essencial em projetos de análise de dados. Ela é importante porque garante a qualidade e a confiabilidade dos dados usados na análise. Durante o tratamento, erros nos dados são identificados e corrigidos, como valores ausentes ou informações inconsistentes. Além disso, os dados são padronizados e organizados de forma consistente, o que facilita a compreensão e a análise.

O tratamento dos dados foi feito com algumas operações simples com a ideia de remover os dados com resultados nulos que se adicionados trarão resultados incorretos para o dashboard. Para isso foi feita a leitura dos dados da planilha com o uso da biblioteca Pandas como pode ser visto na Figura 5.

Figura 5 - Leitura dos dados

```
7   # Leitura dos dados do brasileirão
8   brasileirao = pd.read_excel("../data/brasileirao2023.xlsx")
9
10  brasileirao_all = brasileirao.copy()
11
12  brasileirao_all_not_copy = brasileirao
13
14  # Limpeza dos dados Nulos
15  brasileirao = brasileirao[brasileirao["Score_m"].notnull()]
```

Fonte: O autor (2023).

4.6 ROTINAS DE DATA CIENCE E MACHINE LEARNING

Nesta seção serão apresentados os principais métodos que farão a geração dos dados principais onde serão utilizados e manipulados para o desenvolvimento dos demais métodos.

4.6.1 Classificação

A tabela de classificação do campeonato é um dos principais processamento necessário para a criação de outras informações relevantes sobre o campeonato. Nessa tabela é calculado os pontos, jogos, vitórias, empates, derrotas, gols marcados, gols contra e o saldo dos gols das equipes. A variável ‘tabela_sort’ é a responsável por guardar a tabela calculada conforme a Figura 6.

Figura 6 - Processamento para a classificação

```
times = brasileirao["Mandante"].unique()
tabela = pd.DataFrame()
for time in times:
    vit = emp = der = pro = con = jog = 0
    mandante = brasileirao[brasileirao["Mandante"] == time]
    for indice, partida in mandante.iterrows():
        if partida["Score_m"] > partida["Score_v"]:
            vit += 1
        elif partida["Score_m"] == partida["Score_v"]:
            emp += 1
        else:
            der += 1
        pro += partida["Score_m"]
        con += partida["Score_v"]
        jog += 1
    visitante = brasileirao[brasileirao["Visitante"] == time]
    for indice, partida in visitante.iterrows():
        if partida["Score_v"] > partida["Score_m"]:
            vit += 1
        elif partida["Score_v"] == partida["Score_m"]:
            emp += 1
        else:
            der += 1
        pro += partida["Score_v"]
        con += partida["Score_m"]
        jog += 1
    new_row = pd.DataFrame(
        {
            "Time": [time],
            "J": [jog],
            "V": [vit],
            "E": [emp],
            "D": [der],
            "GP": [pro],
            "GC": [con],
        }
    )
    tabela = pd.concat([tabela, new_row], ignore_index=True)

tabela["P"] = (tabela["V"] * 3) + tabela["E"]
tabela["SG"] = tabela["GP"] - tabela["GC"]
tabela_sort = tabela.sort_values(by=["P", "V", "SG"], ascending=False)
```

Fonte: O autor (2023).

4.6.2 Cluster

A clusterização foi empregada com a finalidade de agrupar as equipes em categorias, cada uma representando os possíveis destinos no campeonato do ano seguinte. Foram delineados cinco grupos: Título, Libertadores, Sul-Americana, Limbo e Rebaixamento. As equipes que compartilham maior semelhança entre si são alocadas nos respectivos grupos. Cada grupo que a equipe pertence foi adicionada na variável ‘tabela’ na coluna ‘cluster’ conforme a Figura 7.

Figura 7 - Coluna de cluster

```
df_data = tabela[["P", "V", "E", "D", "SG"]]
kmeans = KMeans(n_clusters=5, random_state=0).fit(df_data)
tabela["cluster"] = kmeans.labels_
```

Fonte: O autor (2023).

Após a formação dos clusters, optou-se por desenvolver uma previsão para determinar se a equipe permanecerá no mesmo grupo até o final. Para atingir esse objetivo, implementou-se um modelo de regressão logística. Essa informação será armazenada na varável ‘tabela’ na coluna de ‘cluster_pred’, a aplicação do modelo é possível de ser identificada na Figura 8.

Figura 8 - Aplicação de regressão logística

```
df_data = tabela[["cluster", "P", "V", "E", "D", "SG"]]

X_Train = df_data.drop(columns=["cluster"], axis=1)
X_Test = df_data.drop(columns=["cluster"], axis=1)
y_Train = df_data["cluster"]
y_Test = df_data["cluster"]

sc_x = StandardScaler()
X_Train = sc_x.fit_transform(X_Train)
X_Test = sc_x.fit_transform(X_Test)

logreg = LogisticRegression(solver="lbfgs", max_iter=500)
logreg.fit(X_Train, y_Train)
pred_logreg = logreg.predict(X_Test)
pred_proba = logreg.predict_proba(X_Test)

tabela["cluster_pred"] = pred_logreg
```

Fonte: O autor (2023).

Após a formação e previsão dos clusters, foi estabelecida a aplicação para identificar a probabilidade de cada equipe permanecer em cada grupo. Assim, é obtido uma lista de probabilidades atribuída à variável 'lista_proba'. Posteriormente, esses dados foram percorridos para identificar cada cluster e seu valor de probabilidade correspondente, os quais foram incluídos como colunas na variável 'tabela'. Essa lógica é possível ser visualizada na Figura 9.

Figura 9 - Chances de cada grupo

```
lista_proba = pred_proba.tolist()

data = []  # Lista para armazenar os dicionários de dados

index = 0
for proba in lista_proba:
    for i in range(0, len(proba)):
        new_row = {"index": index, "prob": i, "valor": round(proba[i], 4)}
        data.append(new_row)
    index += 1

df_prob = pd.DataFrame(data)  # Cria o DataFrame a partir da lista de dicionários
df_prob = df_prob.pivot_table(
    index="index", columns="prob", values="valor", aggfunc="sum"
)
df_prob = df_prob.reset_index()
df_prob = df_prob.set_index("index")

df_prob = df_prob.rename(
    columns={0.0: "cl_o", 1.0: "cl_1", 2.0: "cl_2", 3.0: "cl_3", 4.0: "cl_4"}
)

tabela = pd.merge(tabela, df_prob, left_index=True, right_index=True)
```

Fonte: O autor (2023).

4.6.3 Regressão

Para gerar as probabilidades de pontos finais do campeonato foi aplicado o modelo de regressão linear. Primeiramente é necessário obter a soma acumulada de todos os pontos das equipes apresentado na Figura 10.

Figura 10 - Soma acumulada dos pontos

```python
# Método para calcular a regressão
def calcula_pontuacao_regressao():
    rodadas = brasileirao["Rodada"].unique()
    times = brasileirao["Mandante"].unique()
    pontuacao = pd.DataFrame()
    data = []  # Lista para armazenar os dicionários de dados

    for rodada in rodadas:
        for time in times:
            resultado = brasileirao[
                (brasileirao["Rodada"] == rodada)
                & (
                    (brasileirao["Mandante"] == time)
                    | (brasileirao["Visitante"] == time)
                )
            ]
            if len(resultado) > 0:
                resultado = resultado.reset_index()
                if resultado["Mandante"][0] == time:
                    if resultado["Score_m"][0] > resultado["Score_v"][0]:
                        pontos = 3
                    elif resultado["Score_m"][0] == resultado["Score_v"][0]:
                        pontos = 1
                    else:
                        pontos = 0
                else:
                    if resultado["Score_v"][0] > resultado["Score_m"][0]:
                        pontos = 3
                    elif resultado["Score_v"][0] == resultado["Score_m"][0]:
                        pontos = 1
                    else:
                        pontos = 0
                new_row = {"time": time, "rodada": rodada, "pontos": pontos}
                data.append(new_row)  # Adiciona o dicionário à lista de dados

    pontuacao = pd.DataFrame(data)  # Cria o DataFrame a partir da lista de dicionários

    pt_pontuacao = pontuacao.pivot_table(
        index="rodada", columns="time", values="pontos", aggfunc="sum"
    )
    pt_pontuacao_cum = pt_pontuacao.cumsum()
    pt_pontuacao_cum = pt_pontuacao_cum.reset_index()

    return pt_pontuacao_cum
```

Fonte: O autor (2023).

Em seguida percorrer estes valores calculando a regressão linear. O valor obtido pela regressão foi adicionado na variável ‘new_row_regressao’ com as colunas de ‘time’ para a sigla da equipe, ‘pontuacao_final’ com o resultado final da pontuação, ‘intercept’ onde se refere ao ponto em que a linha de regressão linear cruza o eixo vertical e ‘slope’ se refere à inclinação da linha de regressão linear. Todo esse processamento é atribuído ao método ‘calcular_regressao’ que ao final retorna um novo *dataframe* ordenado pela pontuação final. Possível de ser observado na Figura 11.

Figura 11 - Calculando a regressão

```
280  def calcular_regressao():
281      pt_pontuacao_cum = calcula_pontuacao_regressao()
282      colunas = pt_pontuacao_cum.columns
283      df_regressao = pd.DataFrame()
284      data_regressao = (
285          []
286      )  # Lista para armazenar os dicionários de resultados de regressão
287
288      for coluna in colunas:
289          if coluna != "rodada":
290              X = pt_pontuacao_cum["rodada"].values.reshape(-1, 1)
291              y = pt_pontuacao_cum[coluna].values.reshape(-1, 1)
292              regressor = LinearRegression()
293              regressor.fit(X, y)
294              A = regressor.intercept_[0]
295              B = regressor.coef_[0][0]
296              x = A + (B * 38)
297              new_row_regressao = {
298                  "time": coluna,
299                  "pontuacao_final": round(x),
300                  "intercept": round(A, 2),
301                  "slope": round(B, 2),
302              }
303              data_regressao.append(new_row_regressao)
304      df_regressao = pd.DataFrame(data_regressao)
305
306      return df_regressao.sort_values(by="pontuacao_final", ascending=False)
```

Fonte: O autor (2023).

4.6.4 Métodos de Tratamentos

Em determinados momentos, foi necessário realizar tratamentos adicionais, como a obtenção do nome da equipe com base em sua sigla e vice-versa. Para facilitar essas operações, foram desenvolvidos dois métodos: 'getNomeTimeFromSigla' Figura 12, que recebe como parâmetro a sigla da equipe.

Figura 12 - Método 'getNomeTimeFromSigla'

```
209  # Método responsável por retornar o nome do time com base na sigla
210  def getNomeTimeFromSigla(sigla):
211      times = getAllTimes()
212
213      index_of_sigla = times.index[times["Sigla"] == sigla].tolist()[0]
214      nome_time = times.loc[index_of_sigla, "Time"]
215
216      return nome_time
```

Fonte: O autor (2023).

Já o método 'getSiglaFromNome' Figura 13, que recebe como parâmetro o nome da equipe. Esses métodos desempenham um papel fundamental ao permitir a correspondência entre informações da equipe, tornando o processo mais eficiente e preciso.

Figura 13 - Método 'getSgilaFromNome'

```
219  # Método responsável por retornar a sigla do time com base no nome
220  def getSiglaTimeFromNome(nome):
221      times = getAllTimes()
222
223      index_of_sigla = times.index[times["Time"] == nome].tolist()[0]
224      sgila_time = times.loc[index_of_sigla, "Sigla"]
225
226      return sgila_time
```

Fonte: O autor (2023).

4.7 VISUALIZAÇÃO E DASHBOARDS

O resultado do projeto foi a criação dos dashboards. Neste capítulo, será apresentado o processo de criação, que abrange desde a definição das tabelas e métricas até a elaboração dos gráficos. Toda a parte visual foi desenvolvida utilizando a biblioteca Streamlit. São separados por dois tipos de visualização, da competição ou de alguma equipe individualmente.

4.7.1 Dashboard Campeonato

Nesta seção será apresentado as principais funções, métodos, modelos e bibliotecas utilizadas para a criação da visualização sobre o campeonato. Estes podem ser vistos na Figura 14.

Figura 14 - Importação das bibliotecas, variáveis e métodos

```python
import streamlit as st
import pandas as pd
import altair as alt
import numpy as np
from streamlit_extras.metric_cards import style_metric_cards
from dados import (
    tabela_sort,
    df_cluster_grupo,
    getNomeTimeFromSigla,
    brasileirao,
    calcular_tabela,
    calcular_cluster,
    calcular_regressao,
    calcula_regressao_meio_campeonato,
    df_chance_cluster,
    brasileirao_all,
    brasileirao_all_not_copy
)
```

Fonte: O autor (2023).

O dashboard do campeonato exibe as principais métricas e resultados da competição. Para organizar as informações em abas, utilizamos o elemento 'tabs' do Streamlit. Conforme ilustrado na Figura 15.

Figura 15 - Abas de informações do campeonato

```
463   # Método utilizado para criar o Dashboard do campeonato
464   def createDashboardCampeonato():
465       (
466           painel_campeonato,
467           classificao_grupo,
468           classificacao_regressao,
469           chances_campeonato,
470           jogos,
471           resultados
472       ) = st.tabs(
473           [
474               "Campeonato e Classificação",
475               "Classificação - Grupo",
476               "Classificação - Previsão",
477               "Chances de Grupos",
478               "Jogos",
479               "Resultados"
480           ]
481       )
482
483       with painel_campeonato:
484           createPainelCampeonato()
485           createTabelaClassificacao()
486       with classificao_grupo:
487           createTableClassificacaoGrupo()
488           createTableCluster()
489       with classificacao_regressao:
490           createAreaRegressao()
491       with chances_campeonato:
492           createTableChanceCluster()
493       with jogos:
494           createTableJogos()
495       with resultados:
496           createEditResultado()
```

Fonte: O autor (2023).

A primeira aba apresenta os dados do campeonato com o elemento ‘metric’ do Streamlit que cria uma visualização de métricas, elas são parte integrante da biblioteca Python 'style_metric_cards'.

O método 'createPainelCampeonato' tem a responsabilidade de gerar o painel de métricas do campeonato. Para isso, a primeira etapa consiste em definir as colunas que representarão as métricas e, em seguida, preencher essas colunas com os valores a serem apresentados, conforme ilustrado na Figura 16.

Figura 16 - Métrica do campeonato

```python
# Método para criar o painel de dados do campeonato
def createPainelCampeonato():
    st.subheader("Dados do campeonato.")
    card_total_jogo, card_total_gols, card_total_ponto, card_media_gol = st.columns(4)
    card_primeiro_colocado, card_time_mais_gol = st.columns(2)
    total_gol = int(brasileirao["Score_m"].sum() + brasileirao["Score_v"].sum())
    total_ponto = tabela_sort["P"].sum()
    time_mais_gol = getNomeTimeFromSigla(
        tabela_sort[tabela_sort["GP"].notnull().sort_values(ascending=False)][
            "Time"
        ].iloc[0]
    )
    primeiro_colocado = getDadoTabelaClassificacao().iloc[0]
    nome_primeiro_colocado = getNomeTimeFromSigla(primeiro_colocado["Time"])
    total_rodada_jogada = (
        brasileirao[brasileirao["Score_m"].notnull()]["Rodada"].unique().max()
    )
    media_gol_rodada = total_gol / total_rodada_jogada

    card_total_jogo.metric(
        "Total de jogos",
        brasileirao[brasileirao["Score_m"].notnull()]["Rodada"].count(),
    )
    card_total_gols.metric("Total de gols", total_gol)
    card_total_ponto.metric("Total de pontos", total_ponto)
    card_media_gol.metric("Média de gol por rodada", round(media_gol_rodada))
    card_primeiro_colocado.metric("Primeiro colocado", nome_primeiro_colocado)
    card_time_mais_gol.metric("Time com mais gols", time_mais_gol)
    style_metric_cards(
        border_left_color="#4fb342",
        background_color="#F0F2F6",
        border_size_px=3,
        border_color="#CECED0",
        border_radius_px=10,
        box_shadow=True,
    )

```

Fonte: O autor (2023).

A tabela de classificação é o resultado dos primeiros passos na coleta dos dados iniciais, sendo também uma das representações mais reconhecidas no contexto do futebol brasileiro. Nela, um cálculo atribui 3 pontos para cada vitória, 1 ponto para empates e nenhum ponto para derrotas, somando-se a outras variáveis dos jogos. Essas informações estão disponíveis no método ‘createTabelaClassificacao’ e de forma detalhada na Figura 17.

Figura 17 - Tabela de classificação

```python
# Método utilizado para criar a tabela de Classificação
def createTabelaClassificacao():
    st.subheader("Classificação Brasileirão 2023.")
    progresso = st.toggle("Progresso dos dados 📈")
    dadoTabelaClassificacao = getDadoTabelaClassificacao()

    iClassificacao = 0
    for i in dadoTabelaClassificacao["Time"]:
        iClassificacao += 1
        dadoTabelaClassificacao.loc[
            dadoTabelaClassificacao["Time"] == i, "Classificação"
        ] = f"{iClassificacao}º"

    dadoTabelaClassificacao = dadoTabelaClassificacao[
        ["Classificação", "Time", "J", "P", "V", "E", "D", "GP", "GC", "SG"]
    ]

    if progresso:
        pon_max = int(dadoTabelaClassificacao["P"].max())
        vit_max = int(dadoTabelaClassificacao["V"].max())
        emp_max = int(dadoTabelaClassificacao["E"].max())
        der_max = int(dadoTabelaClassificacao["D"].max())
        gop_max = int(dadoTabelaClassificacao["GP"].max())
        goc_max = int(dadoTabelaClassificacao["GC"].max())
        sag_max = int(dadoTabelaClassificacao["SG"].max())
        st.dataframe(
            dadoTabelaClassificacao,
            height=750,
            hide_index=True,
            use_container_width=True,
            column_config={
                "P": st.column_config.ProgressColumn(
                    "P", format="%f", min_value=0, max_value=pon_max
                ),
                "V": st.column_config.ProgressColumn(
                    "V", format="%f", min_value=0, max_value=vit_max
                ),
                "E": st.column_config.ProgressColumn(
                    "E", format="%f", min_value=0, max_value=emp_max
                ),
                "D": st.column_config.ProgressColumn(
                    "D", format="%f", min_value=0, max_value=der_max
                ),
                "GP": st.column_config.ProgressColumn(
                    "GP", format="%f", min_value=0, max_value=gop_max
                ),
                "GC": st.column_config.ProgressColumn(
                    "GC", format="%f", min_value=0, max_value=goc_max
                ),
                "SG": st.column_config.ProgressColumn(
                    "SG", format="%f", min_value=0, max_value=sag_max
                ),
            },
        )
    else:
        st.dataframe(
            dadoTabelaClassificacao,
            height=750,
            hide_index=True,
            use_container_width=True,
        )

```

Fonte: O autor (2023).

O diferencial dessa classificação reside na capacidade de visualizar a evolução dos dados, permitindo uma análise da distância necessária para atingir resultados e pontos desejados. Para criar essa progressão, foi empregado o recurso de ‘dataframe’ do Streamlit, configurando as colunas de forma a exibir a trajetória dos dados. Isso proporcionou uma abordagem dinâmica e esclarecedora para acompanhar o desempenho ao longo do tempo. Veja na Figura 18.

Figura 18 - Progressão dos dados

```
    if progresso:
        pon_max = int(dadoTabelaClassificacao["P"].max())
        vit_max = int(dadoTabelaClassificacao["V"].max())
        emp_max = int(dadoTabelaClassificacao["E"].max())
        der_max = int(dadoTabelaClassificacao["D"].max())
        gop_max = int(dadoTabelaClassificacao["GP"].max())
        goc_max = int(dadoTabelaClassificacao["GC"].max())
        sag_max = int(dadoTabelaClassificacao["SG"].max())
        st.dataframe(
            dadoTabelaClassificacao,
            height=750,
            hide_index=True,
            use_container_width=True,
            column_config={
                "P": st.column_config.ProgressColumn(
                    "P", format="%f", min_value=0, max_value=pon_max
                ),
                "V": st.column_config.ProgressColumn(
                    "V", format="%f", min_value=0, max_value=vit_max
                ),
                "E": st.column_config.ProgressColumn(
                    "E", format="%f", min_value=0, max_value=emp_max
                ),
                "D": st.column_config.ProgressColumn(
                    "D", format="%f", min_value=0, max_value=der_max
                ),
                "GP": st.column_config.ProgressColumn(
                    "GP", format="%f", min_value=0, max_value=gop_max
                ),
                "GC": st.column_config.ProgressColumn(
                    "GC", format="%f", min_value=0, max_value=goc_max
                ),
                "SG": st.column_config.ProgressColumn(
                    "SG", format="%f", min_value=0, max_value=sag_max
                ),
            },
        )
```

Fonte: O autor (2023).

Na segunda aba, é disponibilizada a classificação por grupos, que é uma das visualizações essenciais no projeto. É possível verificar as equipes associadas a cada grupo

usando o elemento 'selectbox' do Streamlit, como ilustrado na Figura 19. Essa funcionalidade torna a identificação das equipes em cada grupo conveniente e intuitiva para o usuário.

Figura 19 - Classificação por grupo

```
144  # Método utilizado para criar a tabela de Classificação com Grupo
145  def createTableClassificacaoGrupo():
146      st.subheader("Classificação Brasileirão 2023 por Grupo.")
147      opcao = st.selectbox("Escolha o Grupo", (df_cluster_grupo["grupo"]))
148      classificacaoGrupo = getClassificaoGrupo()
149      st.dataframe(
150          classificacaoGrupo[classificacaoGrupo["Grupo"] == opcao],
151          hide_index=True,
152          use_container_width=True,
153      )
```

Fonte: O autor (2023).

Dentro da mesma aba, também são exibidos os gráficos históricos que destacam a participação das equipes em cada grupo ao longo das rodadas, uma representação visual significativa. Para criar esses gráficos, são utilizados os dados gerados nas etapas iniciais do processo, bem como componentes do Streamlit e da biblioteca Python Altair. Essa integração de ferramentas proporciona uma análise detalhada da evolução das equipes ao longo do torneio.

Para criar essa visualização, foi elaborado o método 'createTableCluster'. Inicialmente, é oferecida a opção de selecionar a rodada por meio do elemento 'slider' do Streamlit, que é usado para processar a exibição do gráfico. Com base na rodada escolhida, o gráfico ilustra a trajetória das equipes entre os grupos. Estes gráficos foram criados com o ‘altari_chart’ da biblioteca Altair, uma referência para criação de gráficos, foram desenvolvidos um total de 4 gráficos, todos com o mesmo objetivo, porém diferentes visualizações. O processo de criação desses gráficos pode ser acompanhado na Figura 20.

Figura 20 - Gráfico dos grupos

```
    grafico_circle_1, grafico_circle_2 = st.columns(2)

    grafico_circle_1.altair_chart(
        alt.Chart(clusters, title="Brasileirao - Gráfico 1")
        .mark_circle(size=200)
        .encode(
            x="Rodada:O",
            y=alt.Y("Time:O", sort=colocacao),
            size=alt.Size("Grupo:O", scale=alt.Scale(domain=domain)),
            color=alt.Color(
                "Grupo:O", scale=alt.Scale(domain=domain, range=range_color)
            ),
            tooltip=[
                alt.Tooltip("Time", title="Time"),
                alt.Tooltip("sum(Pontos)", title="Pontos"),
            ],
        )
    )
```

Fonte: O autor (2023).

Na terceira aba, é apresentada a previsão dos resultados finais do campeonato. Nessa etapa, os dados obtidos pelo método 'calcular_regressao' são integrados ao elemento 'dataframe', e a montagem dessa visualização é conduzida pelo método 'createTabelaRegressao', conforme exmplificado na Figura 21.

Figura 21 - Tabela de regressão

```
# Método utilizado para a criação da tela de previsão dos dados (Regressão)
def createTabelaRegressao():
    dadoTabelaClassificacao = calcular_regressao()

    iClassificacao = 0
    for i in dadoTabelaClassificacao["time"]:
        iClassificacao += 1
        dadoTabelaClassificacao.loc[
            dadoTabelaClassificacao["time"] == i, "Classificação"
        ] = f"{iClassificacao}º"

    dadoTabelaClassificacao = dadoTabelaClassificacao[
        ["Classificação", "time", "pontuacao_final", "intercept", "slope"]
    ]
    st.dataframe(
        dadoTabelaClassificacao,
        height=750,
        hide_index=True,
        use_container_width=True,
        column_config={
            "time": st.column_config.Column("Time"),
            "pontuacao_final": st.column_config.Column("Pontuação"),
        },
    )

```

Fonte: O autor (2023).

Além disso, dentro da mesma aba, os usuários têm a opção de selecionar a rodada que desejam visualizar. Dessa forma, o cálculo de regressão é recalculado com base na rodada escolhida, e uma nova tabela é apresentada, levando em consideração os dados a partir da rodada selecionada.

Na quarta aba, é exibida uma tabela que fornece as probabilidades de cada time pertencer a determinados grupos. Esses dados são derivados do processamento inicial do projeto e são apresentados usando o elemento 'dataframe' do Streamlit. O método responsável por criar essa visualização é o 'createTableChanceCluster', conforme ilustrado na Figura 22.

Figura 22 - Tabela chance de grupo

```
269  # Método utilizado para criar a tabela de chances do time em cada grupo (clustes)
270  def createTableChanceCluster():
271      st.subheader("Chances de grupos.")
272      df_chance_pred = df_chance_cluster().copy()
273      df_chance_pred.rename(
274          columns={
275              "cl_o": "Rebaixamento",
276              "cl_1": "Sul-Americana",
277              "cl_2": "Libertadores",
278              "cl_3": "Limbo",
279              "cl_4": "Título",
280          },
281          inplace=True,
282      )
283      df_chance_pred["Título"] = trataValorPorcentagemTime(df_chance_pred["Título"])
284      df_chance_pred["Libertadores"] = trataValorPorcentagemTime(
285          df_chance_pred["Libertadores"]
286      )
287      df_chance_pred["Sul-Americana"] = trataValorPorcentagemTime(
288          df_chance_pred["Sul-Americana"]
289      )
290      df_chance_pred["Limbo"] = trataValorPorcentagemTime(df_chance_pred["Limbo"])
291      df_chance_pred["Rebaixamento"] = trataValorPorcentagemTime(
292          df_chance_pred["Rebaixamento"]
293      )
294      df_chance_pred = df_chance_pred[
295          ["Time", "Título", "Libertadores", "Sul-Americana", "Limbo", "Rebaixamento"]
296      ]
297      st.dataframe(
298          df_chance_pred.sort_values(by="Título", ascending=False),
299          hide_index=True,
300          height=750,
301          use_container_width=True,
302          column_config={
303              "Título": st.column_config.Column("Título %"),
304              "Libertadores": st.column_config.Column("Libertadores %"),
305              "Sul-Americana": st.column_config.Column("Sul-Americana %"),
306              "Limbo": st.column_config.Column("Limbo %"),
307              "Rebaixamento": st.column_config.Column("Rebaixamento %"),
308          },
309      )
```

Fonte: O autor (2023).

Na quinta aba, são apresentados os jogos, os quais são obtidos a partir da base de dados gerada. Essa visualização foi construída usando o elemento 'text' do Streamlit, com a adição de

filtros através do elemento 'radio' e a capacidade de selecionar a rodada desejada para visualizar os jogos. A criação dessa visualização é gerenciada pelo método 'createTabelaJogos', cujos detalhes podem ser observados na Figura 23.

Figura 23 - Apresentação dos jogos

```
409    # Método criado para exibir os jogos
410    def createTableJogos():
411        selecao = st.radio(
412            "Filtros", ["Por rodada", "Disputados", "Todos jogos"], horizontal=True
413        )
414
415        if selecao == "Todos jogos":
416            brasileirao_all_copy = brasileirao_all.copy()
417            brasileirao_all_copy = brasileirao_all_copy.drop("Temporada", axis=1)
418            rodada = 0
419            for index, row in brasileirao_all.iterrows():
420                if np.isnan(row["Score_m"]):
421                    score_m = "null"
422                    score_v = "null"
423                else:
424                    score_m = int(row["Score_m"])
425                    score_v = int(row["Score_v"])
426                if rodada == 0 | rodada != row["Rodada"]:
427                    rodada = row["Rodada"]
428                    st.subheader(f"")
429                    st.divider()
430                    st.subheader(f"Rodada: {row['Rodada']}")
431                st.markdown(
432                    f"{row['Mandante']}  {score_m}  x  {score_v}  {row['Visitante']}"
433                )
434        elif selecao == "Disputados":
435            brasileirao_copy = brasileirao.copy()
436            rodada = 0
437            for index, row in brasileirao_copy.iterrows():
438                if np.isnan(row["Score_m"]):
439                    score_m = "null"
440                    score_v = "null"
441                else:
442                    score_m = int(row["Score_m"])
443                    score_v = int(row["Score_v"])
444                if rodada == 0 | rodada != row["Rodada"]:
445                    rodada = row["Rodada"]
446                    st.subheader(f"")
447                    st.divider()
448                    st.subheader(f"Rodada: {row['Rodada']}")
449                st.markdown(
450                    f"{row['Mandante']}  {score_m}  x  {score_v}  {row['Visitante']}"
451                )
```

Fonte: O autor (2023).

Quando o login é feito como administrador, é disponibilizada a aba para a edição dos resultados dos jogos, a qual é essencial para a geração de todos os painéis do projeto. Para isso, foi desenvolvido o método 'createEditResultado', que utiliza o elemento 'data_editor' do Streamlit. Esse método é configurado de modo a permitir a edição das colunas 'Score_m' e

'Score_v'. Após efetuar as alterações desejadas, os administradores podem utilizar um botão para salvar as modificações diretamente no arquivo .xlsx. Veja na Figura 24.

Figura 24 - Método de edição dos dados

```
426  # Método para criar a tabela de alteração de resultados
427  def createEditResultado():
428      st.text("Essa ação é disponível somente para Administradores.")
429      dados_editados = st.data_editor(
430          brasileirao_all_not_copy,
431          column_config={
432              "Score_m": st.column_config.NumberColumn(
433                  "Placar Mandante",
434                  help="Gols marcados pelo Mandante",
435                  format="%f",
436                  min_value=0,
437                  max_value=5,
438                  width='medium'
439              ),
440              "Score_v": st.column_config.NumberColumn(
441                  "Placar Visitante",
442                  help="Gols marcados pelo Visitante",
443                  format="%f",
444                  min_value=0,
445                  max_value=5,
446                  width='medium'
447              ),
448          },
449          disabled=["Temporada", "Rodada", "Mandante", "Visitante"],
450          hide_index=True,
451          key='data_editor'
452      )
453      if len(st.session_state["data_editor"]["edited_rows"]) > 0:
454          if st.button('Salvar dados...'):
455              dados_editados.to_excel('../data/brasileirao2023.xlsx', index=False)
```

Fonte: O autor (2023).

4.7.2 Dashboard Time

A seguir será apresentado as principais funções, métodos, modelos e bibliotecas utilizadas para a criação da visualização sobre a equipe individualmente. O dashboard da individual exibe as principais métricas e resultados da mesma no campeonato.

Inicialmente é apresentado uma lista com o elemento ‘selectbox’ para definir qual equipe se deseja analisar. Esse desenvolvimento pode ser visto na Figura 25, após selecionar a equipe é feito todo os cálculos e a montagem do *dashboard* para a equipe.

Figura 25 - Seleção da equipe

```python
# Método responsável pela criação do select com os times
def createSelecboxTime():
    times = getAllTimes()
    st.header("Estatísticas do Time.")
    opcao_padrao = "Selecione um time..."
    times_lista = [opcao_padrao] + times["Time"].tolist()

    selected = st.selectbox("Escolha um time:", times_lista)

    if selected != opcao_padrao:
        createDashboardTime(getSiglaTimeFromNome(selected))

```

Fonte: O autor (2023).

Para manter uma organização nos dados que serão apresentados, foi adicionado uma barra de navegação com o elemento ‘tabs’ do Streamlit. Nessa primeira aba é montado os dados da classificação da equipe com o elemento ‘metric’, estas que são parte integrante da biblioteca Python 'style_metric_cards' e seu desenvolvimento pode ser visto na Figura 26 conforme o método ‘createPainelInfoTime’.

Primeiramente é obtido os dados da tabela de classificação e filtrado para trazer somente sobre o time selecionado, em seguida a montagem das métricas e definido seus valores de acordo com o objetivo de cada métrica.

Figura 26 - Métricas do time

```python
# Método utilizado para criar o painel de informações do time
def createPainelInfoTime(sigla):
    tabela = getDadoTabelaClassificacao()
    index_of_sigla = tabela.index[tabela["Time"] == sigla].tolist()[0]
    st.subheader("Dados da classificação.")
    (
        card_classificacao,
        card_ponto,
        card_jogo,
        card_vitoria,
        card_empate,
        card_derrota,
    ) = st.columns(6)
    card_gol_pro, card_gol_con, card_saldo_gol = st.columns(3)
    card_classificacao.metric(
        "Classificação", f"{tabela.loc[index_of_sigla, 'Classificação']}"
    )
    card_ponto.metric("Pontos", tabela.loc[index_of_sigla, "P"])
    card_jogo.metric("Jogos", tabela.loc[index_of_sigla, "J"])
    card_vitoria.metric("Vitórias", tabela.loc[index_of_sigla, "V"])
    card_empate.metric("Empates", tabela.loc[index_of_sigla, "E"])
    card_derrota.metric("Derrotas", tabela.loc[index_of_sigla, "D"])
    card_gol_pro.metric("Gols Pró", tabela.loc[index_of_sigla, "GP"])
    card_gol_con.metric("Gols Contra", tabela.loc[index_of_sigla, "GC"])
    card_saldo_gol.metric("Saldo de Gols", tabela.loc[index_of_sigla, "SG"])

```

Fonte: O autor (2023).

O método ‘createPainelStatusCampeonato’ foi desenvolvido para mostrar os dados de vitória, empate e derrota, neste foi desenvolvido com o elemento de ‘metric’, ‘bar_chart’ que gera um gráfico de barra e ‘line_char’ para um gráfico de linhas, todos estes elementos são da biblioteca Streamlit.

Conforme a Figura 27, no método “createPainelStatusCampeonato” é obtido os dados dos confrontos resultantes entre as equipes.

Figura 27 - Status da equipe no campeonato

```python
# Método utilizado para criar o painel de status do time dentro do campeonato
def createPainelStatusCampeonato(sigla):
    st.subheader("Status no campeonato.")

    dado_jogos = brasileirao.copy()
    dado_jogo_time_mandante = dado_jogos[dado_jogos["Mandante"] == sigla]
    dado_jogo_time_visitante = dado_jogos[dado_jogos["Visitante"] == sigla]

    dados_mandante = calculaVitoriaDerrotaEmpate(dado_jogo_time_mandante)
    dados_visitante = calculaVitoriaDerrotaEmpate(dado_jogo_time_visitante, True)

    metrica = st.toggle("Métrica")
    st.text("Mandante")
    card_vitorias_mandante, card_derrota_mandante, card_empate_mandante = st.columns(3)
    data_mandante = {
        "Valor": [
            dados_mandante["vitoria"],
            dados_mandante["derrota"],
            dados_mandante["empate"],
        ],
        "Status": ["Vitória", "Derrota", "Empate"],
    }
    st.bar_chart(pd.DataFrame(data_mandante), x="Status", y="Valor", color="#19540F")
    if metrica:
        card_vitorias_mandante.metric("Vitória", dados_mandante["vitoria"])
        card_derrota_mandante.metric("Derrota", dados_mandante["derrota"])
        card_empate_mandante.metric("Empate", dados_mandante["empate"])
    st.text("Visitante")
    card_vitoria_visitante, card_derrota_visitante, card_empate_visitante = st.columns(
        3
    )
    data_visitante = {
        "Valor": [
            dados_visitante["vitoria"],
            dados_visitante["derrota"],
            dados_visitante["empate"],
        ],
        "Status": ["Vitória", "Derrota", "Empate"],
    }
    st.bar_chart(pd.DataFrame(data_visitante), x="Status", y="Valor", color="#19540F")
    if metrica:
        card_vitoria_visitante.metric("Vitória", dados_visitante["vitoria"])
        card_derrota_visitante.metric("Derrota", dados_visitante["derrota"])
        card_empate_visitante.metric("Empate", dados_visitante["empate"])
```

Fonte: O autor (2023).

O cálculo das vitórias empates e derrotas como mandante e visitante é feito com o método ‘calculaVitoriaDerrotaEmpate’ que pode ser observado na Figura 28. Após isso é montado o primeiro gráfico que é o de barra e em seguida o gráfico de linhas e por último as

métricas. Para não mostrar todas as formas de visualizar de uma vez, foi adicionado validações de acordo com o elemento ‘toggle’ estiver ativo, assim é montado uma informação por vez.

Figura 28 - Cálculo para vitória, empate e derrota

```
172  # Método responsável por calcular a quantidade de vitórias ou derrotas do time
173  def calculaVitoriaDerrotaEmpate(dado_jogo, bVisitante=False):
174      retorno = {}
175      retorno["vitoria"] = 0
176      retorno["derrota"] = 0
177      retorno["empate"] = 0
178
179      for index, row in dado_jogo.iterrows():
180          if row["Score_m"] > row["Score_v"]:
181              if bVisitante:
182                  retorno["derrota"] += 1
183              else:
184                  retorno["vitoria"] += 1
185          elif row["Score_m"] < row["Score_v"]:
186              if bVisitante:
187                  retorno["vitoria"] += 1
188              else:
189                  retorno["derrota"] += 1
190          else:
191              retorno["empate"] += 1
192
193      return retorno
194
```

Fonte: O autor (2023).

Para o método ‘createPainelChancesCampeonato’ que desenvolvido para mostrar as chances que a equipe tem de permanecer em cada grupo foi usado os elementos de ‘metric’, ‘bar_chart’ e ‘line_chart’ que fazem parte da biblioteca Streamlit.

Como pode ser visto na Figura 29 os dados a serem exibidos nas métricas são salvos em variáveis e a montagem dos gráficos são validadas com o uso do elemento ‘toggle’, assim cada forma de analisar é disponibilizada ao ativar ou não o *toggle*.

Figura 29 - Chances de grupos

```python
# Método utilizado para criar o painel de Chances do time dentro de campeonato
def createPainelChancesCampeonato(index_of_sigla, df_chance_pred):
    st.subheader("Chances de grupos.")
    rebaixamento = trataValorDashboardTime(df_chance_pred.loc[index_of_sigla, "cl_o"])
    sulAmericana = trataValorDashboardTime(df_chance_pred.loc[index_of_sigla, "cl_1"])
    libertadores = trataValorDashboardTime(df_chance_pred.loc[index_of_sigla, "cl_2"])
    limbo = trataValorDashboardTime(df_chance_pred.loc[index_of_sigla, "cl_3"])
    titulo = trataValorDashboardTime(df_chance_pred.loc[index_of_sigla, "cl_4"])

    (
        card_titulo,
        card_libertadores,
        card_sul_americada,
        card_limbo,
        card_rebaixamento,
    ) = st.columns(5)
    bar_chart = st.toggle("Gráfico de Barra")
    line_chart = st.toggle("Gráfico de Linha")

    data = {
        "%": [rebaixamento, sulAmericana, libertadores, limbo, titulo],
        "Grupos": ["Rebaixamento", "Sul-Americana", "Liberatdores", "Limbo", "Título"],
    }

    card_titulo.metric("Título", f"{titulo}%")
    card_libertadores.metric("Libertadores", f"{libertadores}%")
    card_sul_americada.metric("Sul-Americana", f"{sulAmericana}%")
    card_limbo.metric("Limbo", f"{limbo}%")
    card_rebaixamento.metric("Rebaixamento", f"{rebaixamento}%")
    if bar_chart:
        st.bar_chart(pd.DataFrame(data), x="Grupos", y="%", color="#4fb342", height=500)

    if line_chart:
        st.line_chart(
            pd.DataFrame(data), x="Grupos", y="%", color="#19540F", height=500
        )

```

Fonte: O autor (2023).

Por último o método ‘createPainelRegressaoTime’ é utilizado para montar as métricas com a previsão dos dados finais da equipe, para isso foi utilizado o elementro ‘metric’ e como visto na Figura 30 esses dados são obtidos do método de ‘calcular_regressao’ filtrando somente os dados do time selecionado, assim é obtido os possíveis dados finais do time. Também é buscado o grupo que a equipe estará no final do campeonato, isso é feito com base nos dados da previsão do *cluster que* a equipe vai se encontrar.

Figura 30 - Dados finais da equipe

```python
# Método utilizado para criar o painel de possíveis dados final do time
def createPainelRegressaoTime(sigla):
    dadoTabelaClassificacao = calcular_regressao()
    index_of_sigla = dadoTabelaClassificacao.index[
        dadoTabelaClassificacao["time"] == sigla
    ].tolist()[0]

    iClassificacao = 0
    for i in dadoTabelaClassificacao["time"]:
        iClassificacao += 1
        dadoTabelaClassificacao.loc[
            dadoTabelaClassificacao["time"] == i, "Classificação"
        ] = f"{iClassificacao}"

    dadoTabelaClassificacao = dadoTabelaClassificacao[
        ["Classificação", "time", "pontuacao_final", "intercept", "slope"]
    ]

    st.subheader("Previsão final do time no campeonato")
    card_classificacao, card_ponto, card_grupo = st.columns(3)
    card_classificacao.metric(
        "Classificação",
        f"{dadoTabelaClassificacao.loc[index_of_sigla, 'Classificação']}º",
    )
    card_ponto.metric(
        "Pontos", dadoTabelaClassificacao.loc[index_of_sigla, "pontuacao_final"]
    )
    tabela_regressao_cluster = calcula_regressao_cluster()
    index_of_sigla = tabela_regressao_cluster.index[
        tabela_regressao_cluster["Time"] == sigla
    ].tolist()[0]
    iGrupo = tabela_regressao_cluster.loc[index_of_sigla, "cluster_pred"]
    linha_grupo = df_cluster_grupo.loc[df_cluster_grupo["cluster"] == iGrupo]
    sGrupo = linha_grupo["grupo"].values[0]
    card_grupo.metric("Grupo", sGrupo)
```

Fonte: O autor (2023).

4.8 FUNCIONAMENTO

Nesta parte será detalhado o funcionamento completo do protótipo proposto no projeto, descrevendo suas funcionalidades, a disponibilidade de cada tela e os motivos que justificam sua inclusão, também qual os dados usados para a geração dos *dashboards* e *insights*.

4.8.1 Tela de Login

A tela de login terá o papel de identificar se o acesso será feito por administrador que terá permissões distintas, como por exemplo, alterar os resultados dos jogos e o que se identificar como visitante não será necessário login, assim não poderá fazer alterações, somente visualizar e analisar os dashboards. Para alcançar esse objetivo, foi implementado uma

biblioteca Python denominada *SafeLoader* e, adicionalmente, o *streamlit_authenticator*. A tela pode ser vista de acordo com a Figura 31.

Figura 31 - Tela de login

Fonte: O autor (2023).

Inicialmente será exibido um alerta solicitando o login de administrador, porém caso seja informado os dados incorretos é disparado um alerta diferente, informando que está incorreto como pode ser visto na Figura 32.

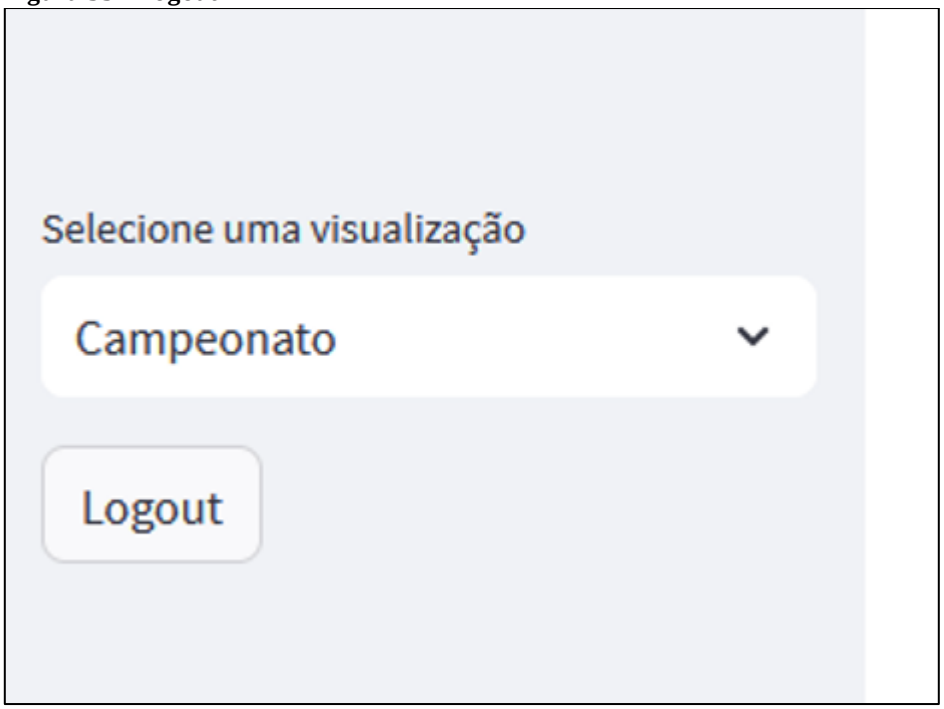
Figura 32 - Login incorreto

Fonte: O autor (2023).

Após efetuar o *login* como administrador, será apresentado os dashboards e suas devidas funcionalidades. Pensando na usabilidade esses dados serão salvos em *cache* do navegador por 30 dias, assim não será necessário informar os dados todas vez. Caso seja necessário realizar

do *logout* existe o botão juntamente ao *sidebar* lateral para realizar a ação de *logout*, veja na Figura 33.

Figura 33 - Logout

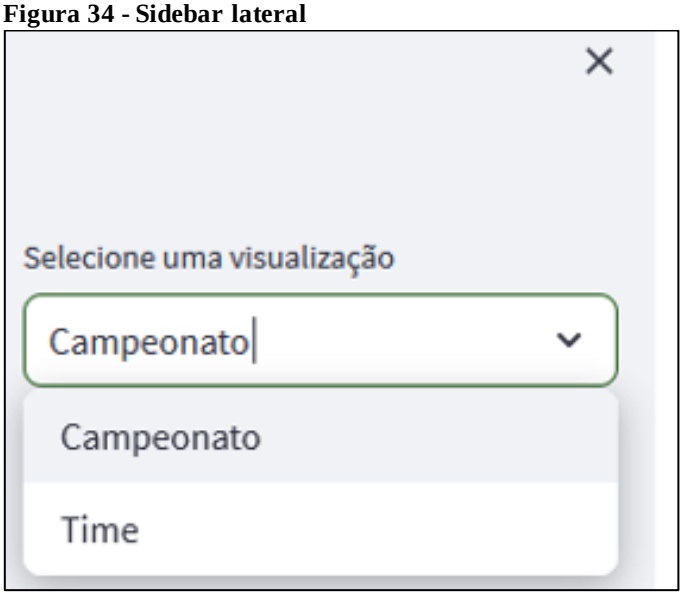


Fonte: O autor (2023).

4.8.2 Sidebar

Imediatamente após o login, os painéis de controle são exibidos. Foi incorporada uma barra lateral à esquerda, que permite selecionar entre o modo de visualização dos dados do campeonato como um todo ou a análise individual dos times. Veja na Figura 34.

Figura 34 - Sidebar lateral



Fonte: O autor (2023).

4.8.3 Visualização e Análise do Campeonato

Para esta seção, será apresentado os elementos utilizados para a criação das visualizações do campeonato.

O elemento inicial é a aba de navegação, que exibe toas as guias contendo informações variadas sobre o campeonato e oferece diversas maneiras de visualizar e analisar os dados relacionados ao campeonato. A separação das informações foi adicionada com o intuito de separa as informações para não gerar confusão ao usuário, pois são muitas visualizações diferentes. A navegação pode ser observada conforme a Figura 35.

Figura 35 - Abas de navegação

Fonte: O autor (2023).

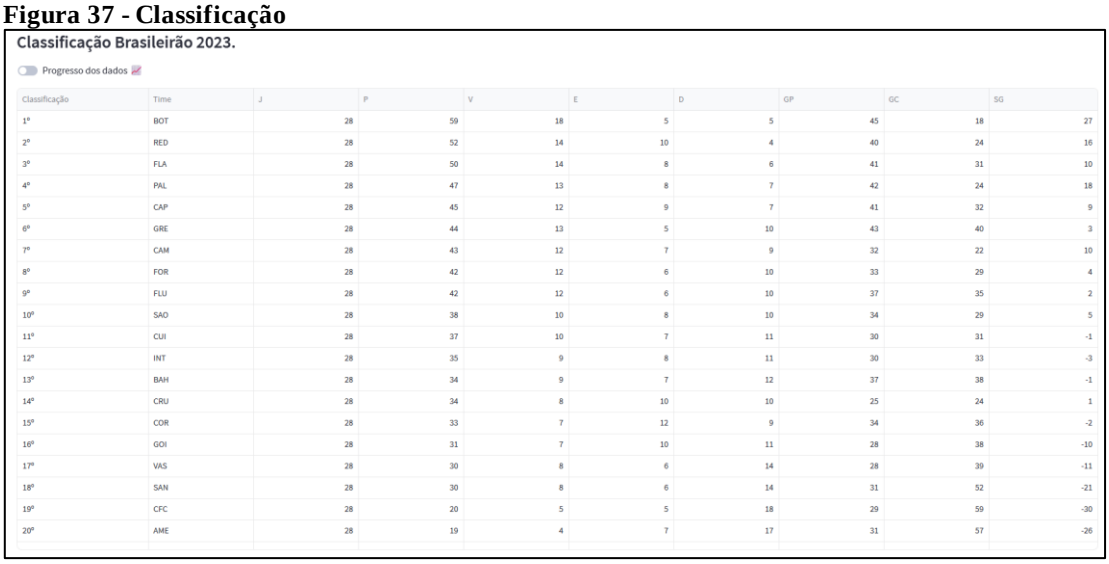
A aba inicial apresentará informações abrangentes sobre o campeonato, incluindo sua classificação geral. Inicialmente, é encontrado métricas relevantes, como o número total de jogos disputados, a quantidade total de gols marcados, a equipe líder do campeonato, a equipe na última posição, bem como aquelas que mais e menos gols marcaram. Essas métricas foram incorporadas com o propósito de fornecer ao usuário uma visão geral dos dados do campeonato. As métricas podem ser vistas na Figura 36.

Figura 36 - Dados do campeonato

Fonte: O autor (2023).

A tabela de classificação é representação mais conhecida e frequentemente consultada pelos usuários interessados no campeonato. Portanto, ela foi desenvolvida e disponibilizada na aba inicial do protótipo. Nessa tabela, são apresentados os seguintes detalhes sobre as equipes: posição na classificação, nomes dos times, número de jogos disputados, pontuação acumulada, vitórias, empates, derrotas, gols marcados, gols sofridos e saldo de gols. Observe a tabela de classificação na Figura 37.

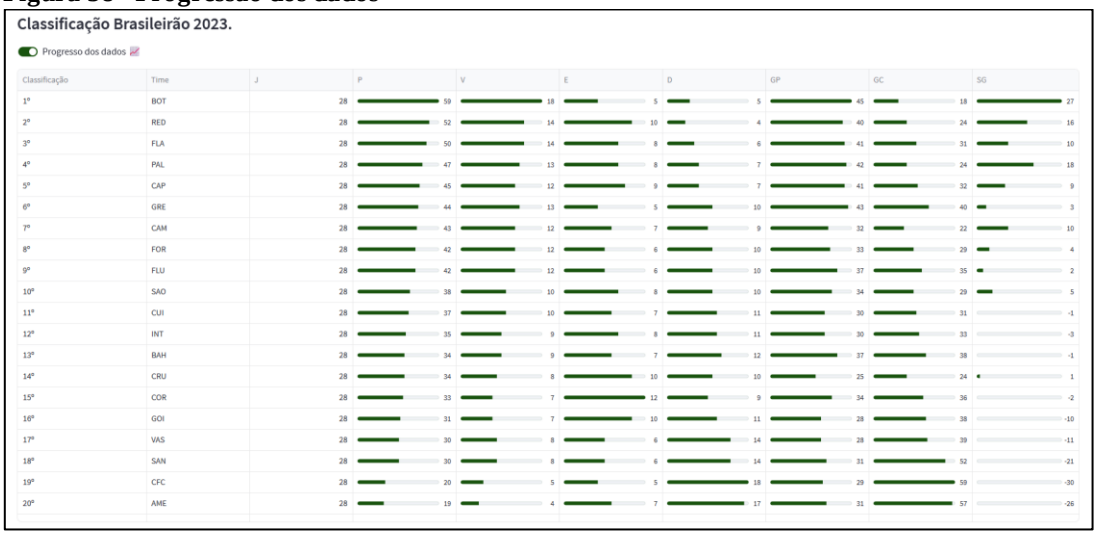
Figura 37 - Classificação



Fonte: O autor (2023).

Ainda na tabela de classificação foi disponibilizado uma flag para representar a progressão dos dados, assim possibilita uma visualização dos dados das equipes. Essa progressão mostra uma barra com o valor atual até o mais alto a distância faltante como pode ser visto na Figura 38.

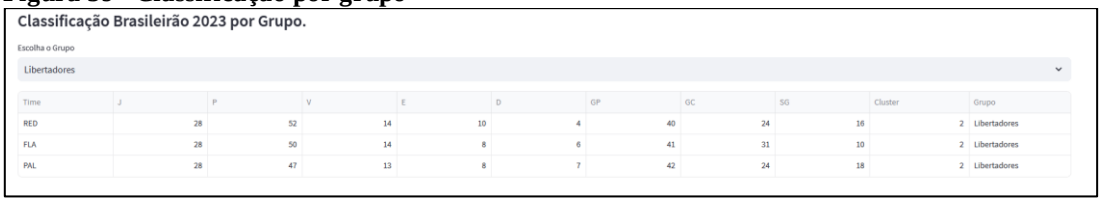
Figura 38 - Progressão dos dados



Fonte: O autor (2023).

A segunda aba de navegação foi criada para oferecer uma alternativa de classificação, organizando os times em grupos com base em seu desempenho no campeonato. Nessa aba, você pode selecionar o grupo que seja visualizar e, ao fazer a escolha, será apresentada uma tabela com os times que fazem parte desse grupo específico. Para uma representação visual, por favor, consulte a Figura 39.

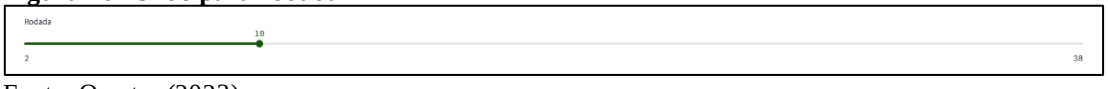
Figura 39 - Classificação por grupo



Fonte: O autor (2023).

Em seguida, é exibido um *slider* que permite ao usuário escolher a rodada que deseja visualizar nos gráficos, como mostrado na Figura 40. Essa funcionalidade foi disponibilizada para que o usuário possa personalizar o que deseja ver.

Figura 40 - Slide para rodada



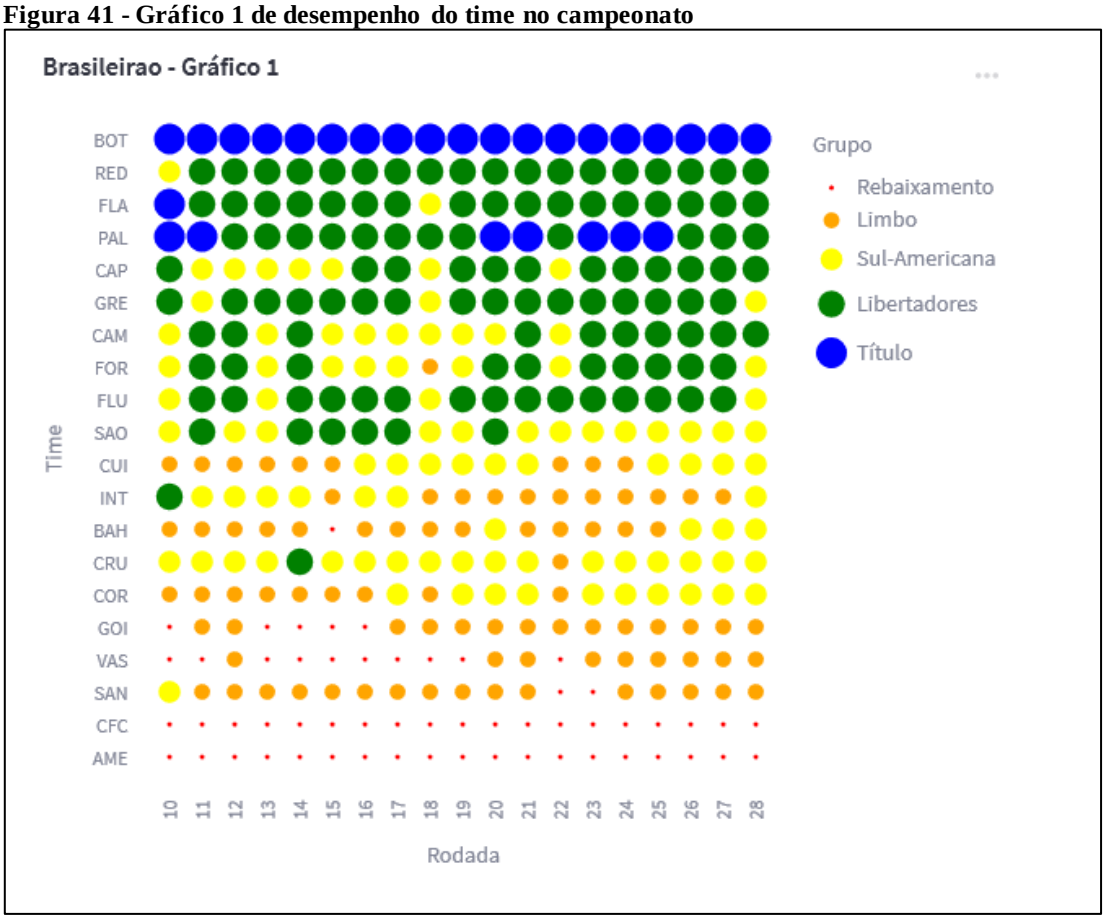
Fonte: O autor (2023).

Por padrão, os valores são apresentados a partir da rodada 10, pois é quando o campeonato começa a se estabilizar. Todos os gráficos contêm as mesmas informações, mas

apresentam de maneiras diferentes no seu eixo X estão as rodadas e no eixo Y os times, também ao passar o mouse é possível verificar a pontuação do time para aquela rodada.

Na Figura 41, as diferenças entre os grupos são indicadas por meio de um símbolo de círculo, com cores e tamanhos diferentes, acompanhados de uma descrição que explica o que cada um deles representa.

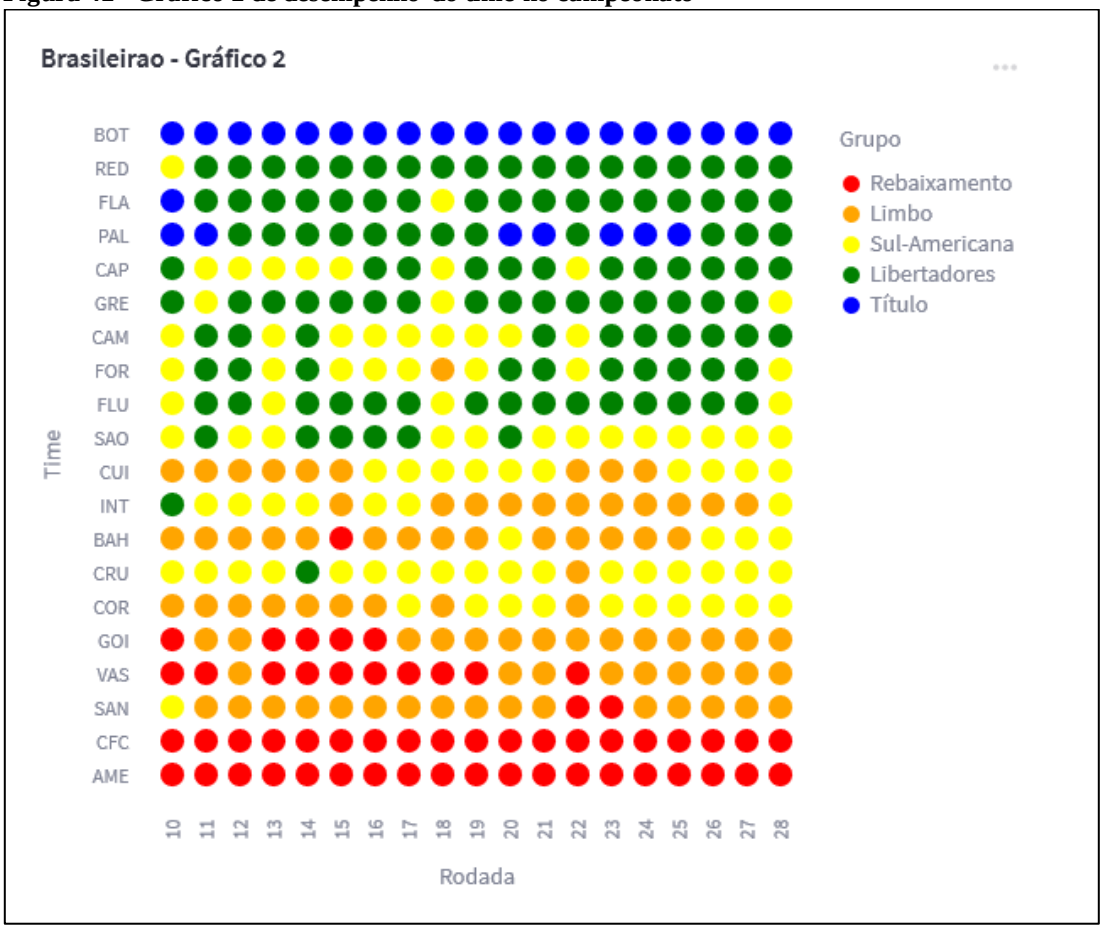
Figura 41 - Gráfico 1 de desempenho do time no campeonato



Fonte: O autor (2023).

Na Figura 42, é exibida uma representação mais simplificada, com apenas símbolos de círculo, cores e uma descrição que esclarece o significado de cada cor.

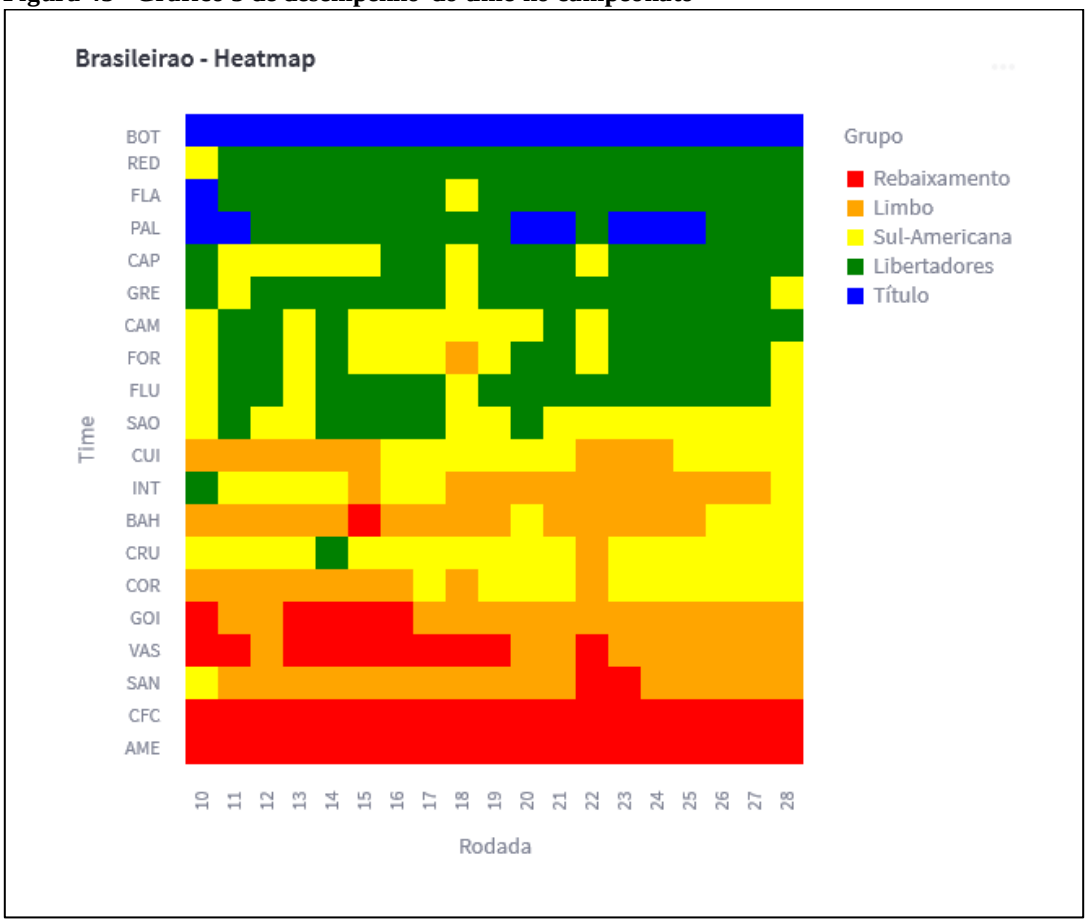
Figura 42 - Gráfico 2 de desempenho do time no campeonato



Fonte: O autor (2023).

Na Figura 43, a representação é ainda mais simplificada, distinguindo-se apenas pelas cores e fornecendo uma descrição para esclarecer o significado de cada uma.

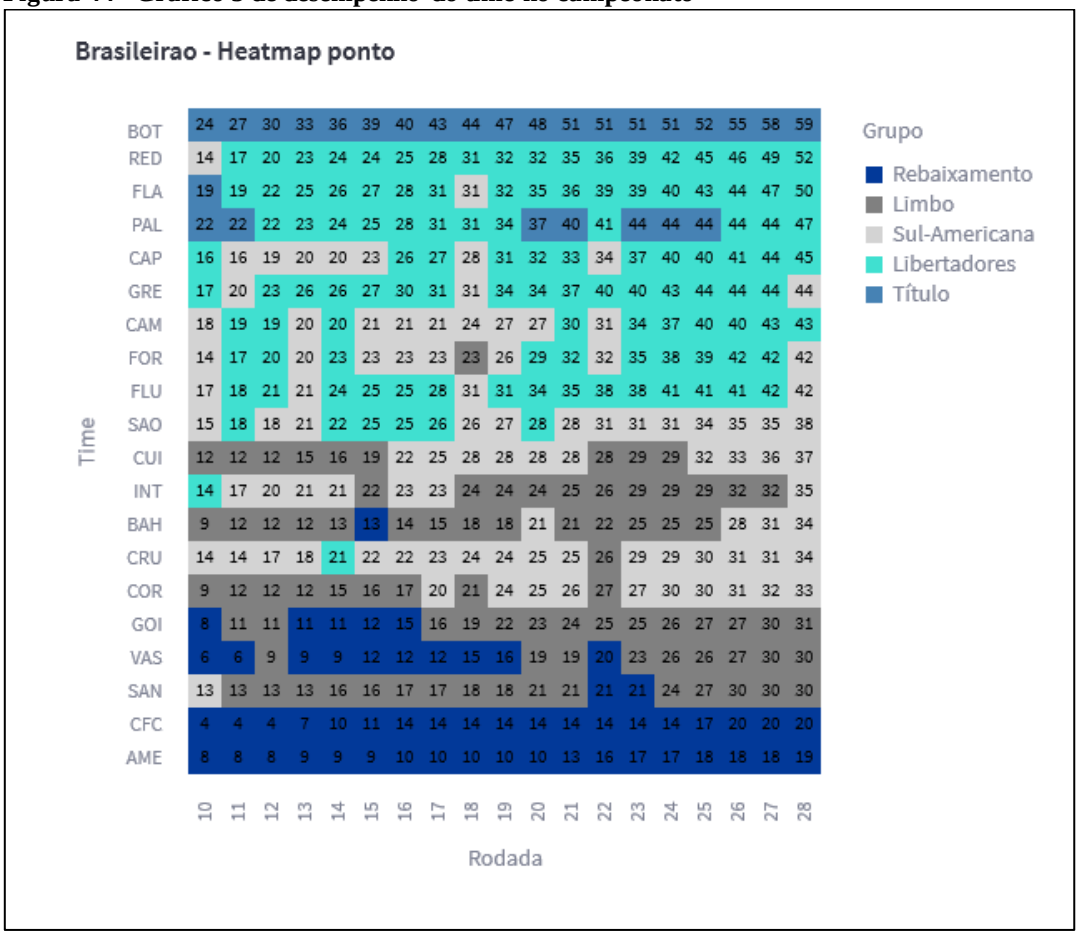
Figura 43 - Gráfico 3 de desempenho do time no campeonato



Fonte: O autor (2023).

No último gráfico, uma apresentação completa é oferecida, com a distinção nas cores dos grupos e a pontuação exibida diretamente, sem a necessidade de passar o mouse sobre o gráfico. Além disso, uma descrição lateral é fornecida para explicar as diferenças entre as cores. Consulte a Figura 44 para uma visualização.

Figura 44 - Gráfico 3 de desempenho do time no campeonato



Fonte: O autor (2023).

Essas diferentes representações foram criadas para permitir que o usuário identifique o desempenho de uma equipe ao longo das rodadas e determine quando ela começou a melhorar ou piorar, além de quando se junto a determinados grupos na competição.

Na próxima aba de navegação, é exibida uma tabela de pontuação final. No entanto, é importante observar que esses dados são estimativos e podem não refletir resultado real do campeonato. A tabela apresenta a classificação, nome das equipes e suas pontuações finais, permitindo que o usuário tenha uma ideia do possível campeão e classificados dos demais campeonatos. Para uma melhor observação, observe a Figura 45.

Figura 45 - Tabela de pontuação final

Tabela de possíveis pontos finais.

Classificação	Time	Pontuação
1º	BOT	83
2º	RED	67
3º	FLA	67
4º	PAL	65
5º	GRE	65
6º	CAP	61
7º	FLU	60
8º	FOR	57
9º	CAM	57
10º	CUI	52
11º	SAO	51
12º	COR	46
13º	INT	46
14º	CRU	45
15º	GOI	42
16º	BAH	41
17º	SAN	39
18º	VAS	37
19º	CFC	28
20º	AME	26

Fonte: O autor (2023).

Para proporcionar uma nova perspectiva nos dados disponíveis, incluímos uma tabela na qual o usuário pode escolher a rodada usada para o cálculo, isso permite verificar quais times se destacam a partir da rodada selecionada. Ao alterar a rodada, todos os dados na tabela são atualizados, proporcionando diferentes cenários de desfecho, como pode ser observado na Figura 46.

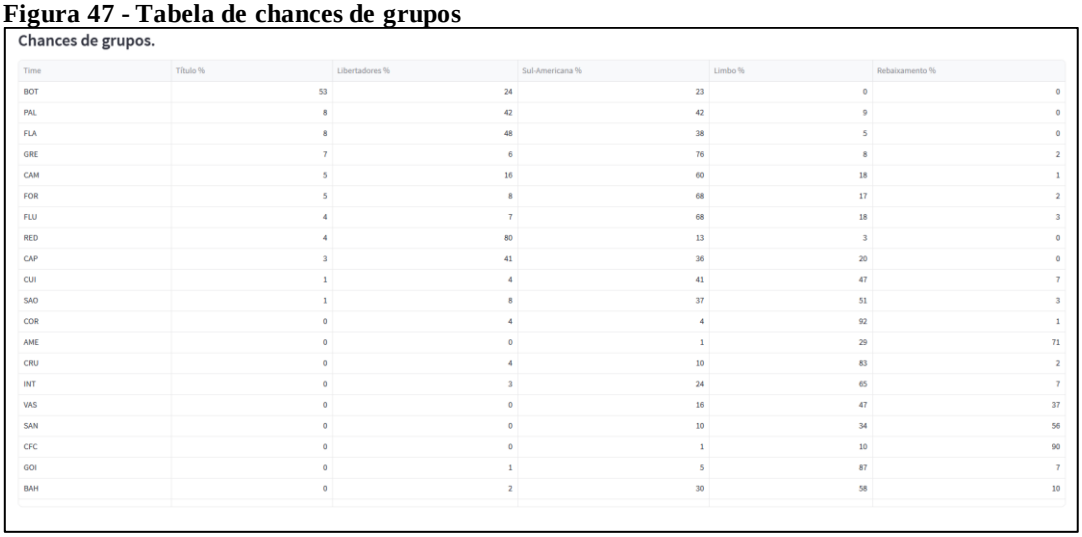
Figura 46 - Tabela de pontuação final com alteração de rodada

Classificação	Time	Pontuação
1°	RED	76
2°	BOT	70
3°	CAM	66
4°	FLA	66
5°	CAP	62
6°	FOR	62
7°	GRE	58
8°	PAL	56
9°	FLU	54
10°	SAO	49
11°	CUI	48
12°	BAH	48
13°	INT	47
14°	VAS	46
15°	SAN	45
16°	CRU	44
17°	COR	43
18°	GOI	40
19°	CFC	30
20°	AME	29

Fonte: O autor (2023).

Na aba seguinte de navegação, é visto uma tabela que mostra as probabilidades de cada equipe permanecer em seu respectivo grupo. A tabela inclui as equipes e grupos, juntamente com as chances expressas em porcentagem, como ilustrado na Figura 47.

Figura 47 - Tabela de chances de grupos



Time	Título %	Libertadores %	Sul-Americana %	Limbo %	Rebaixamento %
BOT	53	24	23	0	0
PAL	8	42	42	9	0
FLA	8	48	38	5	0
GRE	7	6	76	8	2
CAM	5	16	60	18	1
FOR	5	8	68	17	2
FLU	4	7	68	18	3
RED	4	80	13	3	0
CAP	3	41	36	20	0
CUI	1	4	41	47	7
SAO	1	8	37	51	3
COR	0	4	4	92	1
AME	0	0	1	29	71
CRU	0	4	10	83	2
INT	0	3	24	65	7
VAS	0	0	16	47	37
SAN	0	0	10	34	56
CFC	0	0	1	10	90
GOI	0	1	5	87	7
BAH	0	2	30	58	10

Fonte: O autor (2023).

A aba seguinte, mostra os jogos do campeonato, com a possibilidade de aplicar filtros para exibir apenas os jogos desejados. Possibilidade de escolher os jogos por rodada, os já disputados ou todos. Quando definido o filtro, a apresentação é montada, mostrando as equipes envolvidas e seus respectivos resultados, como visto na Figura 48.

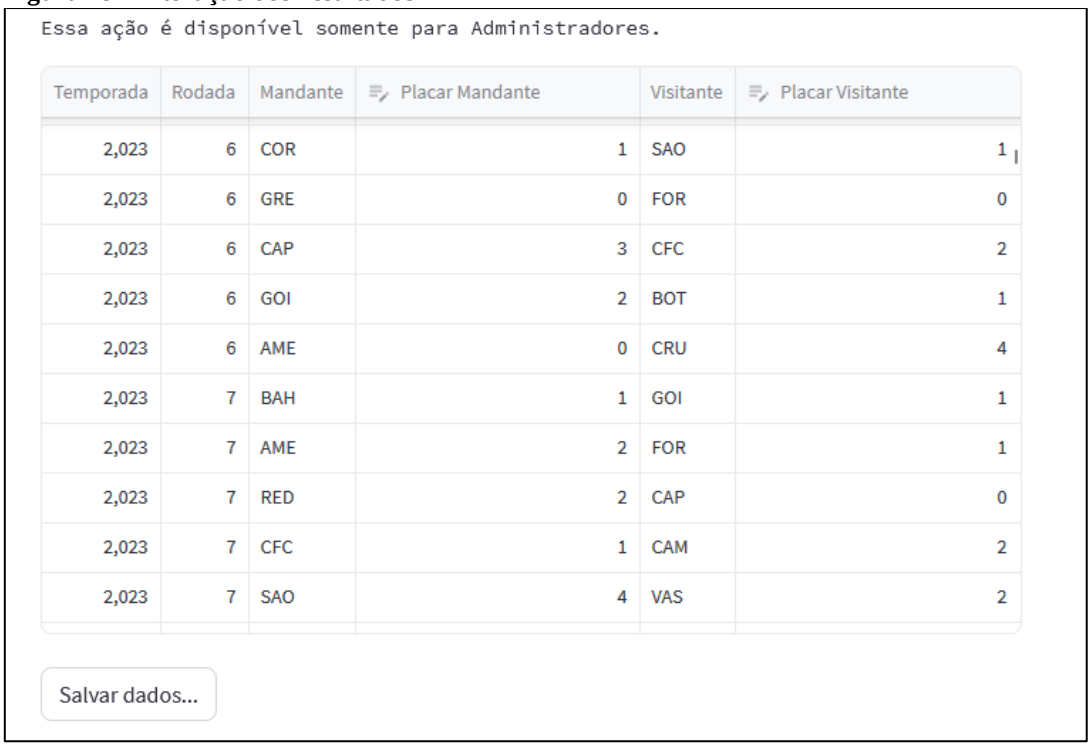
Figura 48 - Jogos

Fonte: O autor (2023).

Para a última aba de navegação, que está disponível apenas para acesso de administrador, é encontrado uma tabela com os resultados dos jogos, que serve como a principal base de dados para a construção dos painéis. Quando realizado alguma alteração nas

informações, um botão estará disponível para salvar os dados, afetando assim todo o protótipo. Verifique a Figura 49 para uma melhor observação.

Figura 49 - Alteração dos resultados



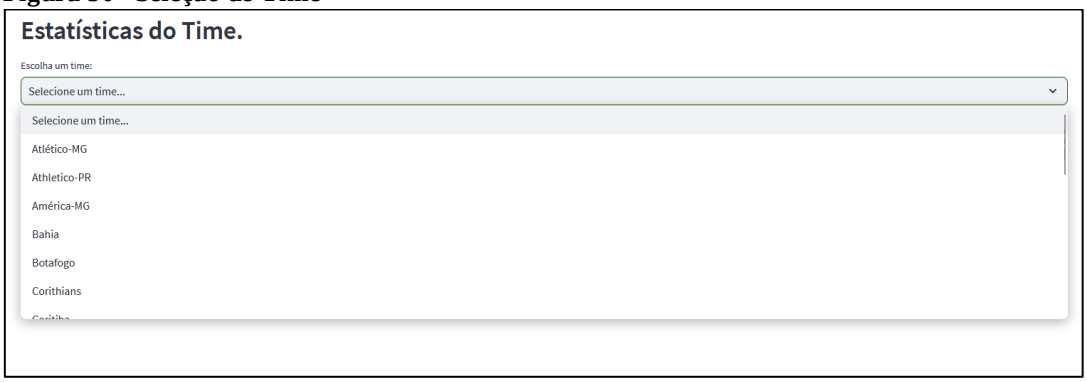
Fonte: O autor (2023).

4.8.4 Visualização e Análise do Time Individualmente

Alterando o modo de visualização para o formato de equipe, resultará em uma abordagem individual dos dados da equipe, ainda com base no campeonato, porém com visualizações específicas. Essas visualizações permitem a apresentação das informações visuais e analíticas referentes à equipe escolhida.

Ao acessar, será disponibilizado uma lista com os times participantes, onde é feita a escolha e da equipe para a apresentação dos *dashboards.* Essa lista pode ser vista na Figura 50.

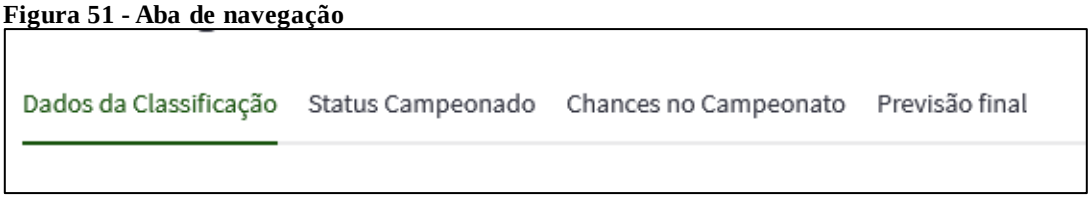
Figura 50 - Seleção de Time



Fonte: O autor (2023).

Após selecionado a equipe desejada, as informações são carregadas. Devido à vasta quantidade de visualizações e dados analíticos disponíveis, eles foram organizados através de uma barra de navegação, conforme ilustrado na Figura 51.

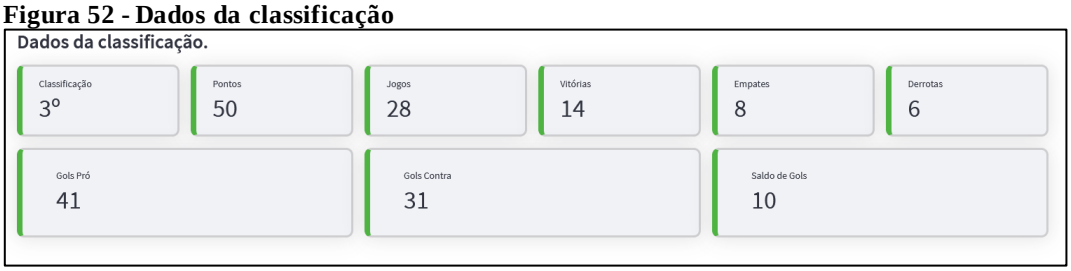
Figura 51 - Aba de navegação



Fonte: O autor (2023).

Logo na primeira aba de navegação é encontrado os dados de classificação da equipe, incluindo sua posição no campeonato, pontuação, número de jogos, vitórias empates, derrotas, gols marcados, gols sofridos e saldo de gols. Essa apresentação foi projetada para oferecer uma visualização mais clara e organizada da equipe selecionada. Observe a Figura 52.

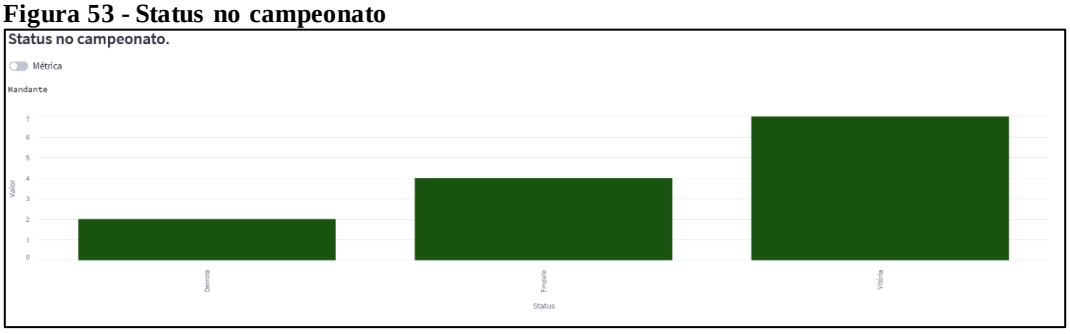
Figura 52 - Dados da classificação



Fonte: O autor (2023).

A próxima aba mostra do status do time no campeonato que praticamente consiste em verificar a quantidade de vitórias, empates e derrotas foram como mandante ou visitante. Para observar isso, foi adicionado um gráfico para observação, onde o eixo X representa o status e Y a quantidade como pode ser visto na Figura 53.

Figura 53 - Status no campeonato



Fonte: O autor (2023).

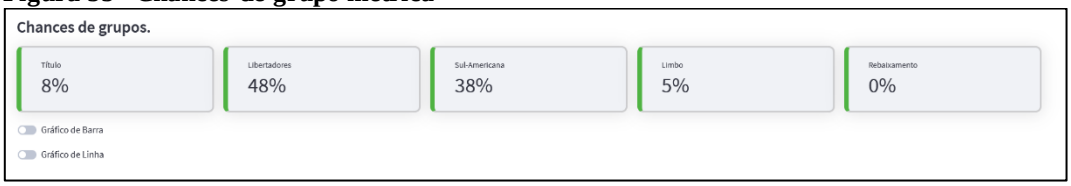
Ainda na observação do status do time no campeonato, foi adicionado a possibilidade de alterar a forma de visualizar, é possível verificar com as métricas, assim disponibilizando diferentes formas de analisar as informações. Verifique a Figura 54.

Figura 54 - Status no campeonato com métrica

Fonte: O autor (2023).

Na terceira aba de navegação foi dedicada para apresentar as chances do time em permanecer nos grupos, inicialmente é apresentada com métricas como visto na Figura 55.

Figura 55 - Chances de grupo métrica

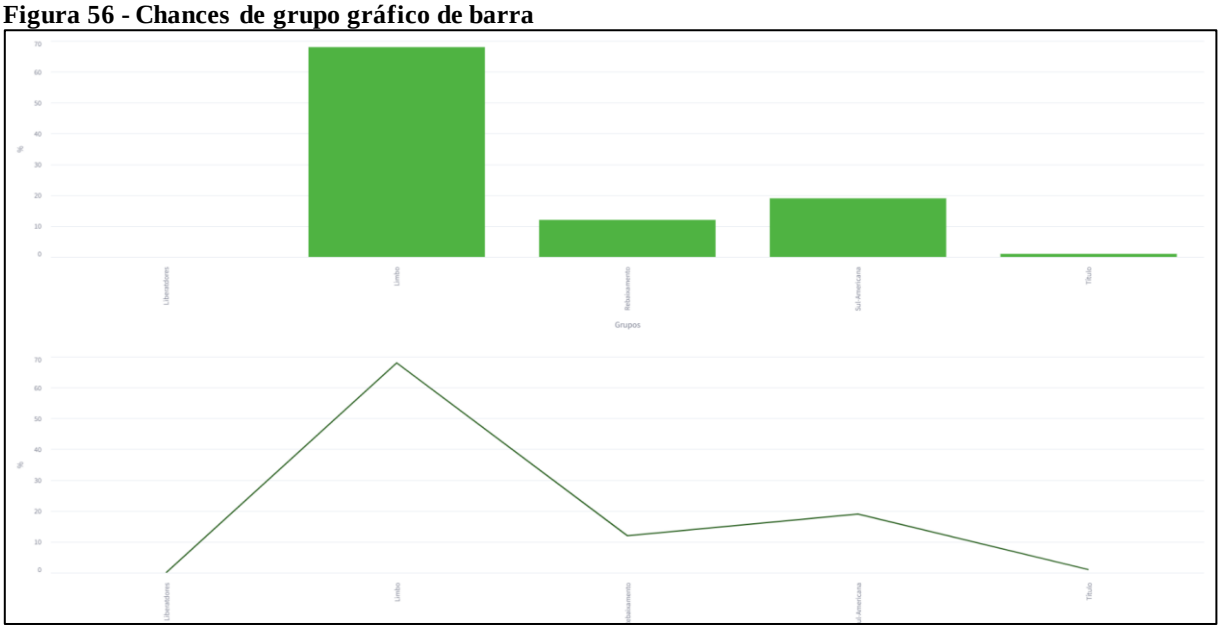


Fonte: O autor (2023).

Para dar mais opções de verificar esses dados e analisar, foi adicionado para selecionar a apresentação em um gráfico de barras ou de linhas como pode ser visto na Figura 56.

Esses dados são disponibilizados para que o usuário possa ver individualmente a chance de a equipe participar de cada grupo, possibilita verificar se ele está perto do título ou rebaixamento, mas também dos demais campeonatos.

Figura 56 - Chances de grupo gráfico de barra



Fonte: O autor (2023).

Para a última aba de navegação, são exibidos os dados potenciais finais da equipe. A apresentação é baseada em métricas que indicam a classificação final, a pontuação e o grupo ao qual a equipe pertence. Isso permite que o usuário tenha uma compreensão clara do desempenho final de sua equipe. Uma melhor apresentação pode ser vista na Figura 57.

Figura 57 - Possíveis dados finais da equipe

Fonte: O autor (2023).

4.9 ENTREVISTA COM USUÁRIOS

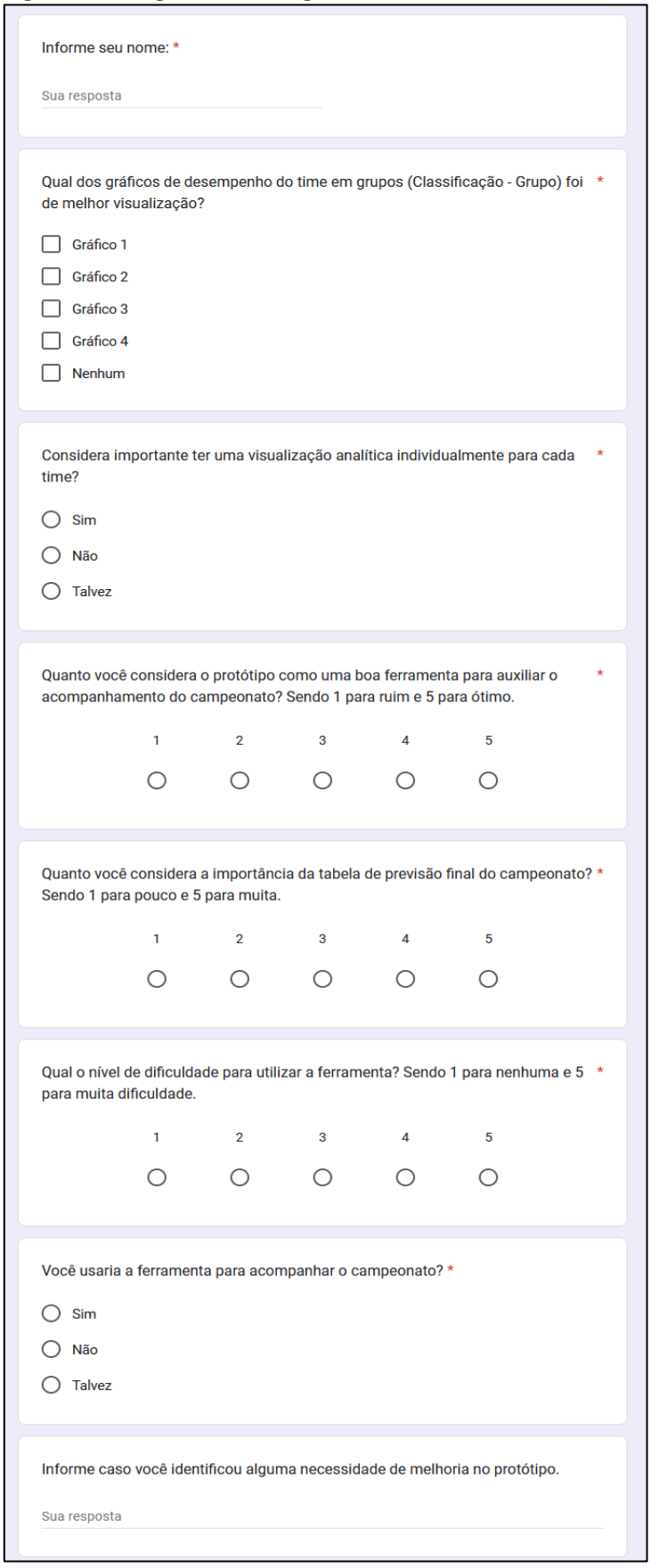
Nesta parte é descrito detalhadamente a execução das entrevistas com os usuários, cujo propósito é obter *feedbacks* fundamentais acerca da utilização da ferramenta desenvolvida neste projeto. Para a obtenção dessas informações, empregou-se a criação de um formulário utilizando a plataforma *Google Forms*. Essa ferramenta proporciona a elaboração de formulário com perguntas e respostas diversas, e ao término do processo de perguntas, gera gráficos com base nas respostas obtidas.

Considerando que o projeto é feito localmente, foi essencial o uso de ferramentas para permitir que os usuários tivessem a oportunidade de experimentar a ferramenta de maneria prática. Para viabilizar essa experiência, recorreu-se ao Ngrok, permitindo o acesso remoto à porta da máquina local onde o projeto está rodando. Além disso, em determinados casos, foram conduzidos testes presenciais diretamente na máquina local. Após a conclusão dos testes e uso da ferramenta, os usuários responderam às perguntas do formulário em questão.

As respostas obtidas desempenham um papel muito importante no processo do protótipo, proporcionando uma base sólida para a análise que auxiliaram a conclusão do projeto. Além disso, esses insights foram valiosos para identificar oportunidade de aprimoramento, contribuindo assim com desenvolvimento futuros da ferramenta.

Todas as perguntas foram feitas de forma objetivar a fim de obter dados de partes principais do projeto, as perguntas podem ser vistas na Figura 58.

Figura 58 - Perguntas do Google Forms



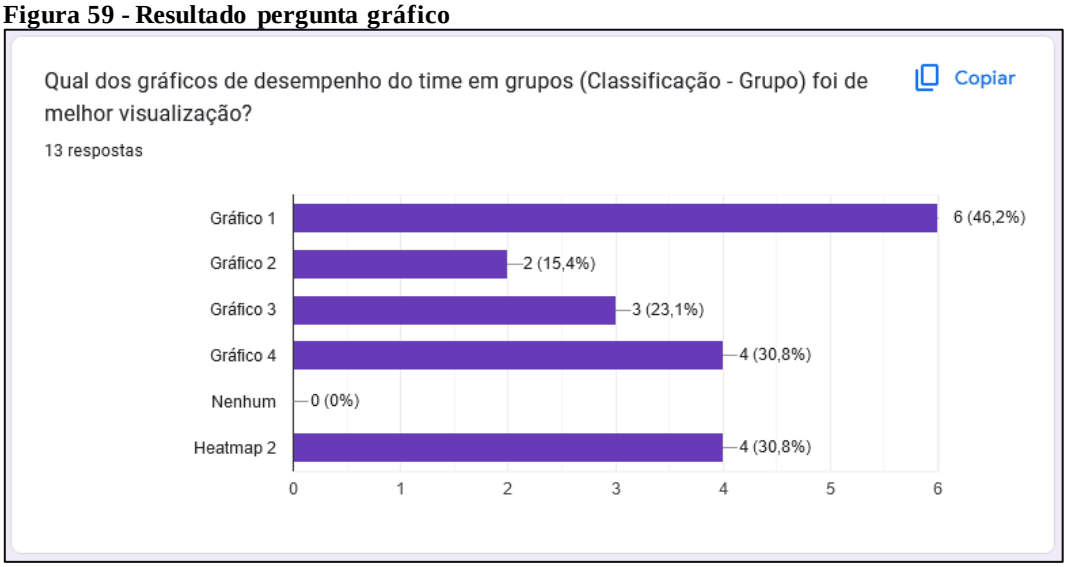
Fonte: O autor (2023).

4.9.1 Resultado da Pesquisa

Após a realização desta entrevista, procedemos à análise dos gráficos gerados, os quais compilaram um total de 13 repostas de usuários. Essa quantidade de *feedback* já proporciona *insights* valiosos para potenciais modificações futuras e avaliações da viabilidade de continuação do protótipo.

A definição de quais gráficos de desempenho do time no campeonato será feita com base nos resultados das respostas vistos na Figura 59, onde o ‘Gráfico 1’ e ‘Gráfico 4’ tiveram predominância.

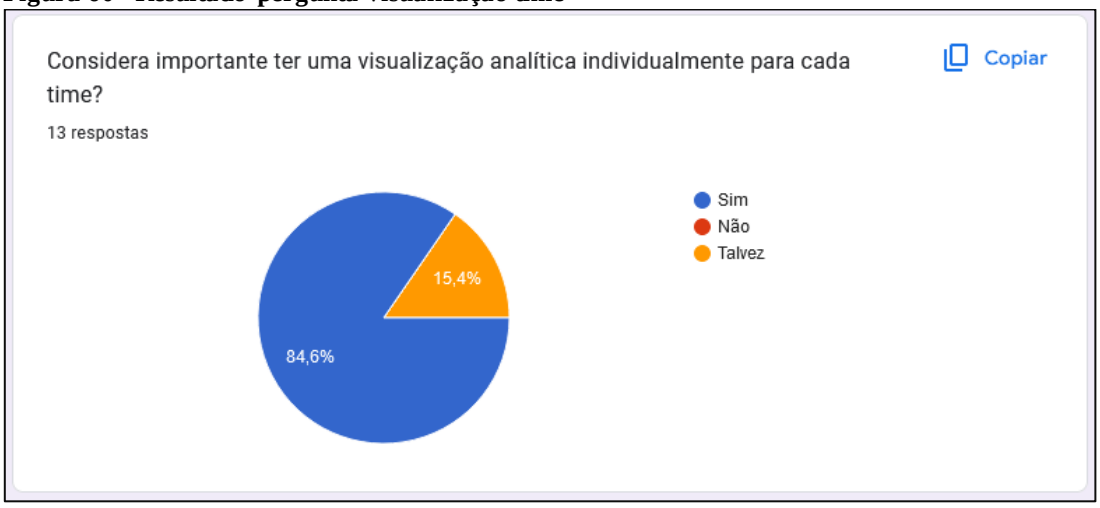
Figura 59 - Resultado pergunta gráfico



Fonte: O autor (2023).

Outro dado importante em vista das perguntas foi sobre a importância da visualização individual de cada time, para uma análise mais específica. Dos 13 entrevistados foi obtido 84,6% como respostas positivas e 15,4% como respostas de incertezas como pode ser observado na Figura 60.

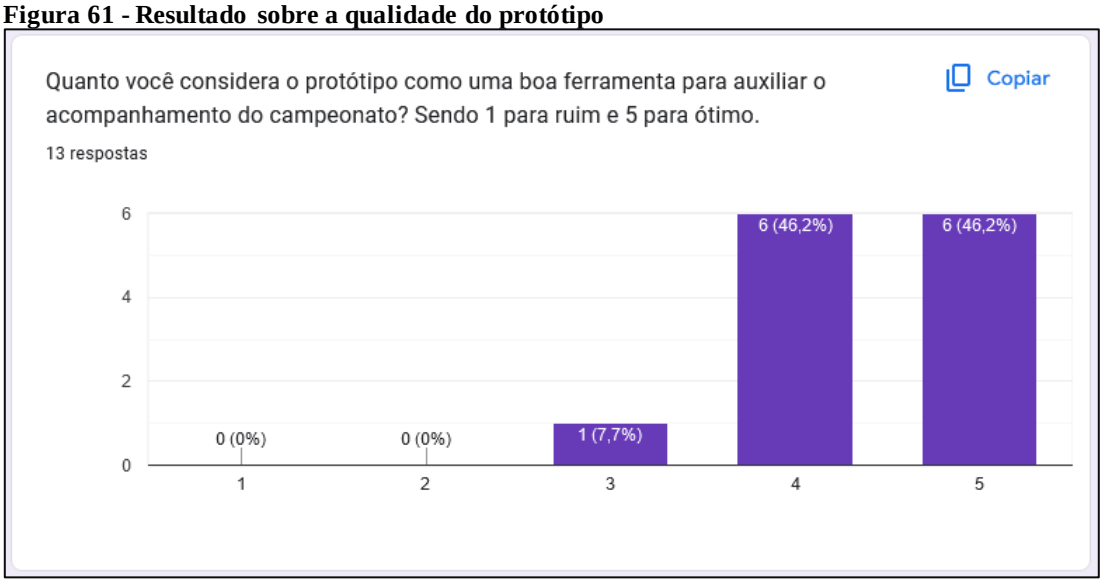
Figura 60 - Resultado pergunta visualização time



Fonte: O autor (2023).

A qualidade do protótipo como uma boa ferramenta para visualização e análise do campeonato também foi questionada com resposta em níveis sendo 1 para ruim e 5 para ótimo. Com a quantidade de entrevistados não se teve resultados negativos, um resultado mediano com total de 7,7% resposta com valor 4 com total de 46,2% e com valor com total de 46,2%. Esses dados podem ser observados na Figura 61.

Figura 61 - Resultado sobre a qualidade do protótipo

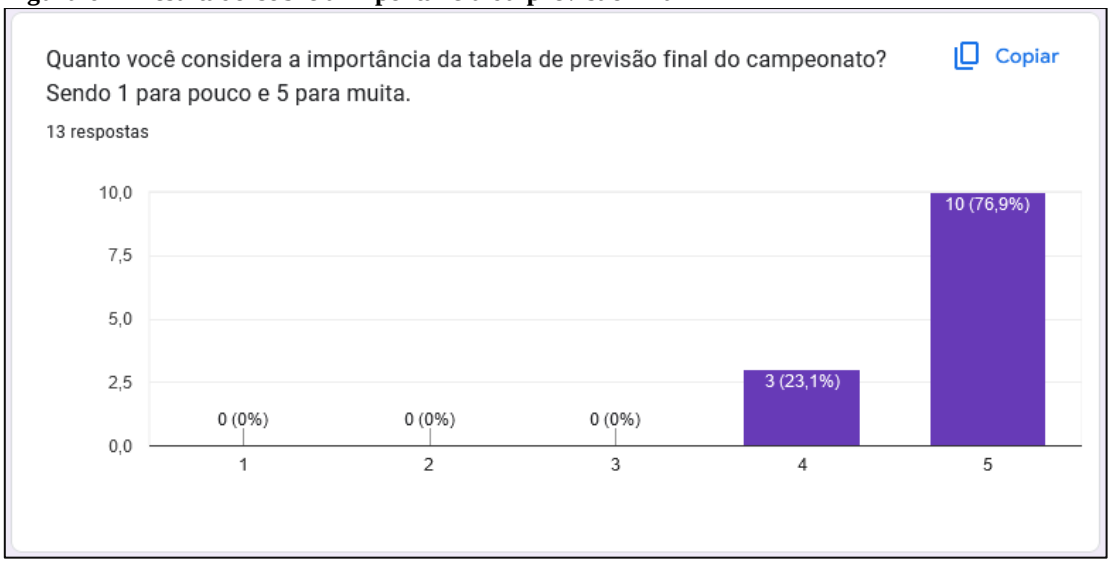


Fonte: O autor (2023).

Outra questão que foi abordada é sobre a importância da tabela de previsão dos pontos finais do campeonato, que foi abordada com resposta em níveis sendo 1 para pouco importante e 5 para muito importante. Os entrevistados julgam de muita importância essa previsão, pois

conforme a Figura 62, pode ser visto o valor 4 com um total de 23,1% e de valor 5 com o restante no total de 76,9%.

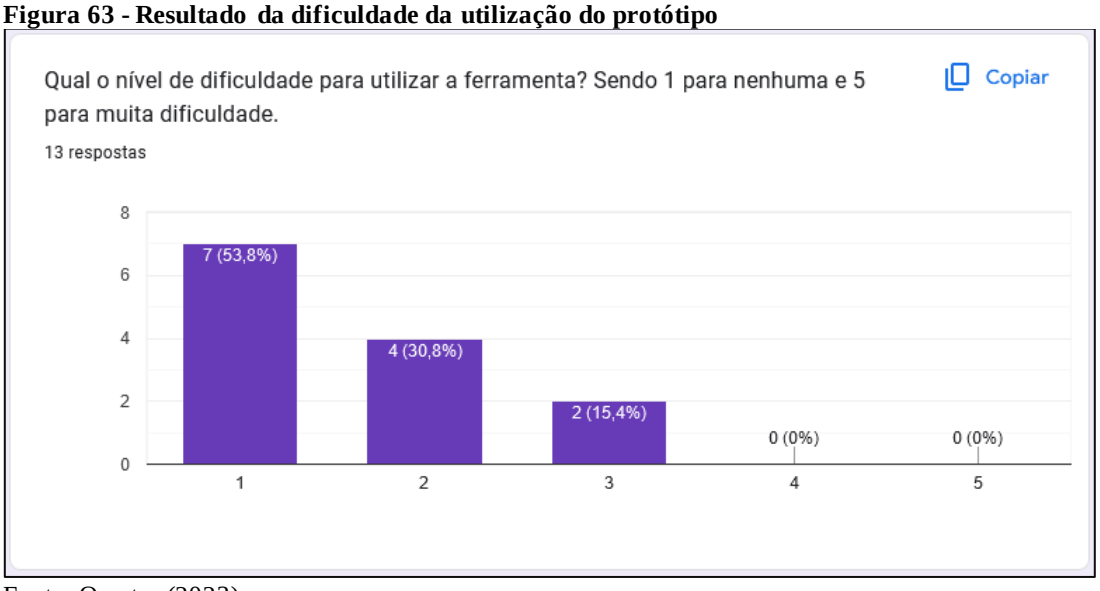
Figura 62 - Resultado sobre a importância da previsão final



Fonte: O autor (2023).

A dificuldade de utilizar o protótipo foi questionado aos usuários entrevistados com respostas em níveis sendo 1 para nenhuma dificuldade e 5 para muita. Com a quantidade de respostas foi possível ver que existe uma pequena dificuldade, porém a maior parte não. O valor 3 teve um total de 15,4% das respostas, o valor 2 um total 30,8% e para o valor 1 um total de 53,8%. Veja na Figura 63.

Figura 63 - Resultado da dificuldade da utilização do protótipo



Fonte: O autor (2023).

Por último foi questionado se o usuário faria a utilização deste protótipo para acompanhar o campeonato, onde a respostas foram positivas tiveram um total de 84,6% e a de incerteza o total de 15,4% como observado na Figura 64.

Figura 64 - Resultado do uso do protótipo

Fonte: O autor (2023).

5. CONCLUSÃO

O presente trabalho de conclusão de curso teve como o principal objetivo a obtenção e a criação de um protótipo de dashboard voltado para a visualização e análise do Campeonato Brasileiro de Futebol.

Ao longo do desenvolvimento deste estudo, explorado o cenário do futebol, um setor que exerce significativa influência no país, seja na economia, cultura ou social. Esta pesquisa proporcionou uma imersão na aplicação de *Data Science* nesse contexto, relevando que, mesmo com volume limitado de informações, foi possível gerar uma ampla variedade de visualização analíticas. Isso, por sua vez, possibilitou aos usuários obterem diferentes perspectivas sobre o campeonato.

As tecnologias empregadas, incluindo ferramentas e linguagem de programação, desempenharam um papel de extrema importância, sendo elementos fundamentais para o êxito do projeto. A linguagem de programação *Python*, juntamente com suas bibliotecas, desempenhou um papel crucial na realização dos cálculos e na construção da estrutura visual por meio de linhas de código.

Os modelos de *Data Science* implementados proporcionaram resultado altamente satisfatórios que possibilitou uma entrega de visualização com previsões e histórico de desempenho das equipes, o que diferencia este protótipo, diferenciando-o de número outros projetos existentes nessa área.

A entrevista com os usuários revelou-se de extrema importância, pois permitiu a identificação de pontos de aprimoramento no protótipo que não foram vistas no desenvolvimento. Além disso, foi obtido valiosos *feedbacks* sobre a relevância desse estilo de acompanhamento, proporcionando *insights* essenciais para dar continuidade ao projeto. Essa abordagem visa alcançar novos usuários em busca de dados intrigantes para visualização do campeonato ou das equipes.

Pode-se afirmar que o desenvolvimento do protótipo foi desafiador por gerar uma quantidade considerável de informações com uma pequena base de dados e também a utilização de uma linguagem de programação com muitas bibliotecas para serem utilizadas.

Desta forma, pode-se concluir que os objetivos específicos propostos para o trabalho foram alcançados, bem como o objetivo geral, contudo, considerando as limitações do projeto, a seguir estão listadas algumas recomendações para trabalhos futuros, incluindo melhorias e novas funcionalidades para o protótipo.

5.1 RECOMENDAÇÕES DE TRABALHOS FUTUROS

Como sugestão principal para a evolução deste projeto, destaca-se a importância em obter informação adicionais sobre as equipes. A verificação de desfalque antes das partidas, a participação em outros campeonatos além do Brasileirão e a existência de jogadores no limite de cartões. Esses dados são cruciais para a elaboração de um Power Ranking, uma vez que afetam muito no desempenho da equipe. A criação desse Power Ranking se mostra muito valiosa, pois com isso será proporcionado uma análise mais precisa das possíveis performances das equipes.

Além disso, uma implementação que enriqueceria ainda mais o projeto consiste na coleta de dados específicos das partidas, como informações sobre escanteios, faltas, cartões, impedimentos, público presente e rendimento. A inclusão desses dados permitirá a criação de diversas visualizações analíticas adicionais, proporcionando uma compreensão mais aprofundada do campeonato e das dinâmicas de cada equipe.

Outra alteração necessária será a conclusão do sistema de login, incorporando um botão de acesso para visitantes. Isso eliminará a necessidade de entrar como administrador nos protótipos, permitindo a definição simplificada de algumas permissões essenciais, como por exemplo, a alteração dos resultados.

Com base nas entrevistas com os usuários, foi possível identificar algumas áreas que necessitam aprimoramento. Uma pergunta específica, elaborada com o propósito de obter informações de melhoria do protótipo, revelou a necessidade de incorporar os brasões dos times e ajustar a coloração ao selecionar uma equipe. Essas melhorias visam aprimorar significativamente a experiência do usuário às suas expectativas.

REFERÊNCIAS

ALPAYDIN, E. ***Introduction to Machine Learning.*** 2. ed. Cambridge, MA: MIT Press, 2010.

ALTAIR. ***Altair Documentation.*** 2019. Disponível em: https://altair-viz.github.io/ . Acessado em: 05 de nov. de 2023.

ALURA. Pandas: **O que é, para que serve e como instalar.** Disponível em: https://www.alura.com.br/artigos/pandas-o-que-e-para-que-serve-como-instalar. Acessado em 25 de mai. de 2023.

ANALYTICS VIDHYA. ***What Is Python Used For? Features, Applications, and Industries. In:*** *Analytics Vidhya*. Disponível em: https://www.analyticsvidhya.com/blog/2020/01/what-is-python-used-for-features-applications/. Acesso em: 06 mai. de 2023.

AWARI. Streamlit: **A Biblioteca Python que Revoluciona o Desenvolvimento de Aplicações Web.** Disponível em: https://awari.com.br/streamlit-python/?utm_source=blog. Acessado em 25 de mai. de 2023.

AWS. **O que é regressão logística?** Disponível em: https://aws.amazon.com/pt/what-is/logistic-regression/#:~:text=A%20regress%C3%A3o%20log%C3%ADstica%20%C3%A9%20uma,resultados%2C%20como%20sim%20ou%20n%C3%A3o. Acessado em 05 de nov. de 2023

CARVALHO, Vinícius. **PostgreSQL – Banco de Dados para aplicações web modernas.** São Paulo: Casa do Código, 2017. E-book

CBF. **Campeonato Brasileiro de Futebol – Série A – 2023.** Disponível em: https://www.cbf.com.br/futebol-brasileiro/competicoes/campeonato-brasileiro-serie-a/2023 Acessado em 25 de mai. de 2023.

COSTA, Neto - **Estatística**. São Paulo: Edgard Blucher LTDA, 2018.

CRUZ, Felipe. **Python – Escreva seus primeiros programas.** São Paulo: Casa do Código, 2015. E-book

DATACAMP. ***What Is Python Used For? In:*** *DataCamp Community*. Disponível em: https://www.datacamp.com/community/tutorials/what-is-python-used-for. Acesso em: 06 mai. de 2023.

DATA SCIENCE. **Qual a diferença entre as variáveis independentes e dependentes?** Disponível em: https://datascience.eu/pt/matematica-e-estatistica/qual-e-a-diferenca-entre-as-variaveis-independentes-e-dependentes/. Acessado em 25 de mai. de 2023.

FARIAS, M.B. **Data Science: Conceitos, técnicas e aplicações para a tomada de decisões baseadas em dados.** São Paulo: Novatec, 2020.

GOLDSCHIDT, Ronaldo; PASSOS, Emmanuel. ***Data Mining***. Elsevier Brasil. Edição Kindle.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. ***Deep learning*****.** MIT Press, 2016.

HAN, J.; KAMBER, M.; PEI, J. ***Data Mining: Concepts and Techniques***. 3rd ed. Waltham: Morgan Kaufmann, 2011.

HARVE. Numpy: **O que é, vantagens e tutorial inicial.** Disponível em: https://harve.com.br/blog/programacao-python-blog/numpy-python-o-que-e-vantagens-e-tutorial-inicial/. Acessado em 25 de mai. de 2023

HAVERBEKE, Marjin. ***Eloquent JavaScript – A Modern Introduction to Programming*****.** San Francisco: No Strach Press. E-book

IDC. ***Worldwide Machine Learning Software Market to Grow at a CAGR of 43.6% Through 2023, According to IDC***. Disponível em: https://www.idc.com/getdoc.jsp?containerId=prUS45194019. Acessado em 11 de mai. de 2023.

ITECNOLOGIA. **Vantagens e Desvantagens da Linguagem Python**. Disponível em: https://ltecnologia.com.br/blog/vantagens-e-desvantagens-da-linguagem-python/. Acessado em 25 de mai. de 2023.

JAMES, G.; HASTIE, T.; TIBSHIRANI, ***R. Regression Models. In:*** *An Introduction to Statistical Learning: With Applications* in R. New York: Springer, 2013.

JOLLY, Kevin. ***Machine learning with Scikit-Learn quick start guide***. *Packt Publishing Limited*, 2018. E-book

MITCHELL, T. M. ***Machine Learning***. McGraw Hill, 1997. E-book

NETO, Pedro Luiz de Oliveira Costa. **Estatística**. Editora Edgard Blücher Ltda., 2002. E-book.

NGROK. ***Ngrok Documentation***. 2023. Disponível em: https://ngrok.com/docs/. Acessado em 13 de nov. de 2023.

PANDAS. ***Pandas Documentation***. 2021. Disponível em: https://pandas.pydata.org/docs/. Acesso em: 06 mai. de 2023.

PEP8. **PEP 8 – Style Guide for Python Code.** Disponível em: https://peps.python.org/pep-0008/. Acessado em 06 de mai. de 2023.

PETERS, T. ***The Zen of Python***. 2004. Disponível em: https://www.python.org/dev/peps/pep-0020/. Acesso em: 06 de mai. de 2023.

PROVOST, F.; FAWCETT, T. ***Data Science for Business:*** *What You Need to Know about Data Mining and Data-Analytic Thinking. O'Reilly Media*, Inc., 2013.

PYTHON. ***About Python***. Disponível em: https://www.python.org/about/apps/. Acesso em: 06 mai. de 2023.

REAL PYTHON. ***What Can You Do with Python? In:*** *Real Python*. Disponível em: https://realpython.com/what-can-i-do-with-python/. Acesso em: 06 mai. de 2023.

STAGGEMEIER, Michel. **Quero Aprender Dashboard**. 2023 . E-book

STREAMLIT. ***Streamlit Documentation.*** 2023. Disponível em: https://docs.streamlit.io/. Acessado em: 06 mai. de 2023.

VAN ROSSUM, G. et al. **PEP 8 -- Style Guide for Python Code. Python Software Foundation**, 2001. Disponível em: https://www.python.org/dev/peps/pep-0008/. Acesso em: 25 de mai. de 2023.

TRIOLA, Mario F. **Introdução à estatística. 11. ed**. Rio de Janeiro: LTC, 2010.

VALENTE, Marco Tulio. **Engenharia de software moderna**: princípios e práticas para desenvolvimento de software com produtividades. Amazon, 2020. E-book.

VANDERPLAS, Jake. ***Python Data Science Handbook:*** *Essential Tools for Working with Data***.** Estados Unidos: O'Reilly Media, 2016.

FM2S. **Estatística Descritiva Básica e Centralidade.** Disponível em: https://www.fm2s.com.br/blog/estatistica-descritiva-basica-e-centralidade. Acessado em 24 de maio de 2023.

ANEXOS