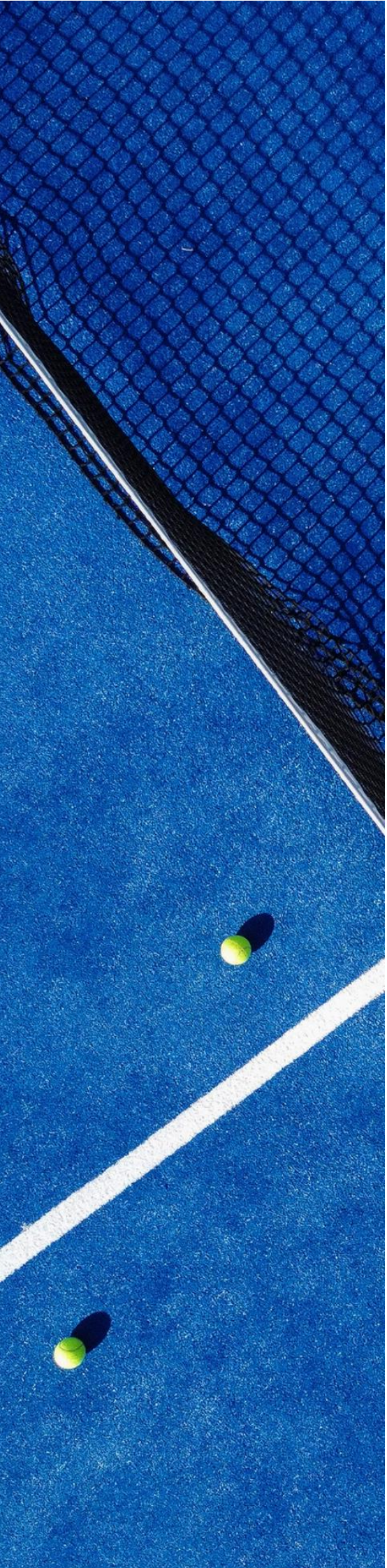


INSTITUTO UNIVERSITÁRIO DE LISBOA

PROJETO APLICADO EM CIÊNCIA DE DADOS I – RELATÓRIO

LICENCIATURA EM CIÊNCIA DE DADOS

Base de Dados ATP Players - Brasil

Grupo 5:

Eliane Susso Efraim Gabriel, Nº 103303

Marco Delgado Esperança, Nº 110451

Maria João Ferreira Lourenço, Nº 104716

Umeima Adam Mahomed, Nº 992397

Docentes: Diana Aldea Mendes, Sérgio Moro

2 de junho de 2023

Índice:

1. Introdução

As competições ATP são torneios de ténis muito prestigiados a nível global, ocorrendo em diversos países. A Confederação Brasileira de Ténis solicitou os nossos serviços de consultoria para desenvolver um modelo capaz de prever o número de sets para a conclusão de um jogo de ténis profissional, com o objetivo de auxiliar os treinadores, jogadores e equipas de apoio a otimizarem os seus treinos e planos de jogo com base no número de sets estimado, e consequentemente, a devida adaptação de estratégias e táticas a serem utilizadas em diferentes cenários.

Para este projeto foi utilizada a metodologia CRISP-DM, que consiste em 6 etapas principais: **compreensão do problema**, **compreensão dos dados**, **preparação dos dados**, **modelação**, **avaliação** e **implementação**.

2. Compreensão do problema

A fim de atender à solicitação da Confederação Brasileira de Ténis de prever o número de sets para a conclusão de um jogo de ténis profissional no Brasil para que os jogadores e treinadores consigam adaptar os planos de treino e táticas de jogo, começamos por perceber o contexto do problema e a origem do *dataset*, uma vez que este pode condicionar toda a formulação e resolução do problema. Esta fase é essencial para garantir resultados confiáveis e úteis ao desenvolver o modelo preditivo.

Dataset ATP

O *dataset* presente no ficheiro *atpplayers.json* contém dados que foram recolhidos do site do ATP e contém informações acerca dos jogos singulares de 10361 jogadores masculinos, pertencentes ao Top 500, que jogaram de 28/03/1973 a 14/02/2022. É de notar que este *dataset* contém diversos jogos realizados em diversos pontos do mundo, pelo que para estudar em concreto o Brasil, tivemos de filtrar os jogos apenas realizados em cidades brasileiras.

ATP Tour

Antes de mais, fomos investigar em que consistia a Associação de Tenistas Profissionais (ATP), que é a entidade responsável pelos circuitos de ténis profissionais masculinos, o ATP Tour, ATP Challenger Tour e ATP Champions Tour.

Esta começou em setembro de 1972, devido à ação de Donald Dell, Jack Kramer e Cliff Drysdale a fim de proteger os interesses dos tenistas profissionais. A denominação desta organização foi alterada ao longo dos anos, mas por fim, retomou ao nome original que prevalece atualmente.

A organização está sediada em Londres, tendo as suas filiais noutros continentes: a filial das Américas localiza-se em Ponte Vedra Beach no estado norte americano da Flórida, a Europa tem a sua filial no Mónaco e a internacional que cobre Ásia, África e Australásia em Sidney, na Austrália. Esta é gerida pelo atual diretor executivo Massimo Calvelli e uma equipa de seis membros.

Os torneiros mais prestigiosos na modalidade desportiva ténis são os de Grand Slam, ou *majors*, que são quatro, Australian Open, Roland Garros (ou French Open), Wimbledon e US Open, que não são sob a alçada de ATP, no entanto, são atribuídos pontos do *ranking* ATP.

A fim de que o público compreendesse melhor a pontuação atribuída a cada competição, os torneios estão classificados em ATP 250 (que concede 250 pontos ao vencedor), ATP 500 (500 pontos) e ATP Masters 1000 (1000 pontos). Ou seja, o sistema de pontuação e o subsequente ranking dos atletas fica claro na nomenclatura da competição. Além disso, os quatro Grand Slams concedem 2000 pontos para o vencedor. Sendo assim, o jogador de ténis recebe uma pontuação conforme o seu desempenho nos campeonatos citados anteriormente.

ATP no Brasil

Relativamente aos torneios que ocorrem no Brasil, o mais famoso é o Rio Open, que teve início em 2014 e ocorre anualmente, apesar de não ter ocorrido em 2021, devido à pandemia. Este é considerado o maior torneio de ténis da América do Sul e pertence ao ATP 500, logo é um torneio bastante prestigiado a nível internacional, onde participam os grandes nomes do ténis a nível mundial.

Também podemos mencionar o Brasil Open que se realizou de 2001 a 2019, apenas para atletas masculinos e ocorreu na Costa do Sauipe, São Paulo e Rio de Janeiro e pertencia ao ATP Tour 250 e o ATP São Paulo que ocorreu de 2001 a 2012, também só para atletas masculinos, sendo substituído pelo Rio Open.

As competições do tipo BRA vs. X são jogos que ocorrem na Davis Cup, nomeadamente na World Group Play-Offs, onde jogadores de países diferentes se defrontam.

Por fim, podemos referir as competições que fazem parte da *ITF Men's Circuit*: Brazil Fx - apenas para atletas masculinos e estão divididas em diferentes níveis consoante o nível dos atletas e o prémio, sendo principalmente para os jogadores ganharem experiência e melhorar os seus *rankings* -, M15 São Paulo - só para atletas masculinos, cujo prémio é de $15000 e ocorre em São Paulo, permitindo o melhoramento dos rankings e a qualificação para o *ATP Challenger Tour* e *ATP World Tour* - e M25 Rio de Janeiro - com as mesmas características que a última, só que o prémio é de $25000 e ocorre no Rio de Janeiro.

Impacto do Ténis na economia, cultura e desenvolvimento do desporto no Brasil

Como referido acima, o Brasil tem competições de ténis a nível nacional e internacional de grande relevância, que trazem para o país treinadores e atletas com renome na modalidade e um grande conjunto de entusiastas e fãs do desporto, o que resulta na geração de um grande nível de receita para o país.

Relativamente a alguns nomes do ténis brasileiro que ganharam títulos do circuito ATP de 1973 a 2022, podemos apontar Gustavo Kuerten, Thiago Monteiro, Julio Goes, Thiago Alves e Ricardo Mello.

Os jogos são cobertos pelos meios de comunicação e muitas vezes transmitidos na televisão, permitindo a chegada ao povo brasileiro e aos cidadãos de outros países de forma fácil. Desta forma, é possível a divulgação do talento nacional, permitindo que os jovens talento mostrem as suas capacidades, compitam em níveis mais prestigiados de futuro e alcancem a desejada internacionalização.

Questões de negócio

Enquanto consultores especializados em análise desportiva, vamo-nos focar na previsão do número de sets necessários para o término de um jogo de ténis no Brasil. Para isto, serão analisados padrões históricos dos jogos que decorreram em solo brasileiro, tendo em conta a forma de jogar dos jogadores, por exemplo a mão que utilizam para jogar, o país de origem dos jogadores sob a esfera climática, considerando que as particularidades do clima brasileiro podem influenciar a condição física dos jogadores se estes não estiverem habituados a este tipo de clima e o tipo de chão que também pode influenciar

os resultados. Além disso, serão analisados em maior detalhe o perfil dos jogadores, os seus rankings e experiência prévia e compará-lo-emos com o seu oponente a fim de perceber se isso tem alguma influência no desempenho e consequente resultado do jogo. Pode-se ainda analisar se a pressão e o cansaço afetam o desempenho dos jogadores, por exemplo, ao final de um ano com mais jogos jogados.

A análise de todas estas componentes fornecerá informações relevantes que poderão certamente ajudar os treinadores e jogadores a adaptarem a sua estratégia de treino e táticas utilizadas durante o jogo com determinado oponente.

Objetivos da análise de dados

A análise dos dados pretende que se encontrem padrões ou relações entre os dados para que se consiga prever de maneira mais precisa e exata o número de sets necessários para a conclusão de um jogo de ténis profissional no Brasil. Iremos analisar em detalhe as variáveis que o *dataset* já nos fornece, criaremos outras variáveis a partir destas e ainda outras que pensamos ser relevantes, mas que não se encontrem nem possam ser construídas a partir dos dados já existentes. Deste modo, esperamos conseguir ter *insights* sobre o desempenho dos jogadores, considerando uma diversidade de variáveis e fatores, tendo em conta a complexidade do tema em estudo.

Hipóteses iniciais

Relativamente às conclusões que serão retiradas futuramente, assumimos que a forma como os jogadores jogam poderá influenciar a forma como o jogo se desenrola criando logo desde o início um desequilíbrio no jogo, ficando um jogador em maior vantagem que o outro, por exemplo, jogadores que joguem com mãos dominantes diferentes. Os jogadores que venham de climas mais frios, se não se conseguirem adaptar ao clima brasileiro, ou que vejam para o evento desportivo muito próximo da data podem ter um desempenho pior por não estarem adaptados às condições atmosféricas particulares da cidade onde se vai desenrolar o torneio e não estarem na melhor das condições físicas. É esperado que os jogadores mais experientes sejam mais velhos e que estejam em posições mais elevadas do *ranking* pelo menos durante os chamados anos dourados, nos quais ainda se encontram numa ótima condição física. Jogadores mais inexperientes e possivelmente mais suscetíveis à pressão podem ter pior desempenho do que o esperado em rondas decisivas.

3. Compreensão dos dados

Nesta etapa começamos por importar o *dataset atpplayers.json* e seguidamente exploramos as diversas variáveis e os respetivos significados presentes no mesmo.

Importação dataset

Importamos o *dataset* presente no ficheiro *atpplayers.json* - sendo que o vamos explorar em maior profundidade de seguida - através da função *read_json* do Pandas. Ao longo do nosso estudo e consoante as necessidades verificadas no decorrer do projeto, também importamos outros ficheiros csv presentes na *internet* com o objetivo de aprofundar o estudo do *dataset* que nos foi atribuído, nomeadamente, *worldcities.csv*, *player_overviews_unindexed_csv.csv*, *Player.csv* e *atpplayers_web_scrapped.csv*. Estes serão explorados na secção "Preparação dos dados", onde explicaremos em detalhe qual a motivação que nos levou a procurá-los e qual foi o uso que lhes demos.

Variáveis em estudo

Relativamente às variáveis presentes no *dataset*, estas podem ser divididas em dois subgrupos: as que estão relacionadas com os jogadores e as que estão relacionadas com os jogos. Para além das que já tínhamos no *dataset* inicial, também vamos explicar as que foram criadas ao longo do projeto, incluindo as que foram criadas e eliminadas logo de seguida, que apenas serviam para validação ou verificação de uma situação em específico que após ter sido avaliada já não fazia sentido mantê-la, com o objetivo de desenvolver um modelo que melhor se adequasse à complexidade do objetivo de negócio.

Relativamente às variáveis já existentes podemos mencionar:

- **_id:** dicionário com identificador único de cada linha, que foi eliminado, após a verificação da não existência de duplicados;
- **PlayerName:** referente ao jogador da partida que aparece em primeiro lugar na base de dados, que vamos considerar como principal para efeitos de compreensão;
- **Born:** país e /ou cidade em que o jogador principal nasceu;
- **Height:** altura do jogador principal em cm;
- **Hand:** indica a mão dominante do jogador principal, que pode ser Right-Handed, se for destro, Left-Handed se for canhoto ou Ambidextrous se ambidestro, ou seja, tem duas mãos dominantes; e quantas mãos utiliza para executar o serviço, pode ser One-Handed Backhand, se usar uma, Two-Handed Backhand, se usar 2, ou Unknown Backhand, se não se souber;

- **LinkPlayer:** link que nos remete para o perfil detalhado do jogador principal no website do ATP Tour;
- **Tournament:** nome do torneio, no qual o jogo teve lugar;
- **Location:** país e/ou cidade, onde determinado torneio se realizou;
- **Date:** indica a data de início e fim de determinado torneio;
- **Ground:** tipo de terreno em que o torneio foi realizado, pode ser Hard - composto por materiais uniformes e endurecidos, com uma camada superficial de acrílico - Grass - relva natural-, Clay - pequenos pedaços de pedra pouco consolidada ou que se parte facilmente e pode ser de cor vermelha ou verde - ou Carpet - materiais sintéticos, como relva artificial com preenchimento de areia;
- **Prize:** valor do prémio monetário, na moeda do país onde se realiza o torneio;
- **GameRound:** fase do torneio a que pertence o jogo, pode pertencer aos jogos de qualificação - quando tem Round Qualifying na designação -, e os de eliminação que têm Round of X ou Finals na designação. Para esta última categoria temos também a Round Robin que é um formato das eliminatórias, em que os jogadores são separados em grupos e jogam todos contra todos e apenas os que passarem a prova vão para as semifinais;
- **Oponent:** nome do jogador que defronta o jogador principal, que com o objetivo de melhor compreender o problema designaremos por jogador oponente;
- **GameRank**, mais tarde apelidado de **RankOpponent:** refere-se ao rank do jogador oponente;
- **WL:** resultado do jogo, é W - se ganhou -, L - se perdeu ou W/0 se o jogador ganha por desistência;
- **Score:** resultados do jogo por sets. Este representa-se sempre por pares, sendo que um par corresponde a um set. O primeiro número refere-se ao número de jogos que o jogador principal daquele jogo venceu e o segundo ao número de jogos que o oponente venceu.

Vamos explicar de seguida, as variáveis que criamos com o objetivo de melhor explorar o *dataset* e de conseguirmos um modelo com maior poder de previsão:

- **DateStart:** data de início do torneio;
- **DateEnd:** data de fim do torneio, que pode coincidir com a data de início do mesmo, no caso em que apenas existe uma data disponível na coluna Date;
- **RankPlayer:** rank do jogador principal num determinado ano;

- **NumberSets:** contagem do número de pares presentes na variável Score, sendo que quando existe RET, ou seja, o jogador desistiu, optamos por não contabilizar estes casos como a ocorrência de um set;
- **_id_str:** transformação da variável _id, em string. Esta foi eliminada após a verificação da não existência de duplicados;
- **City:** cidade presente na variável Location, correspondente à primeira parte desta variável;
- **Country:** segunda parte da variável Location, corresponde ao país onde se realizou o torneio.
- **BornCity:** cidade de naturalidade do jogador principal, correspondente à primeira parte da variável Born;
- **BornCountry:** país de naturalidade do jogador principal, corresponde à segunda parte da variável Born;
- **L_OR_R:** informa se o jogador é destro, esquerdino ou ambidestro e corresponde à primeira parte da variável Hand;
- **BornOpponent:** informação acerca do país e cidade de nascimento do jogador oponente;
- **BornCityOpponent:** cidade de nascimento do jogador oponente;
- **BornCountryOpponent:** país de nascimento do jogador oponente;
- **HeightOpponent:** altura do jogador oponente, em cm;
- **HandOpponent:** mão dominante do jogador oponente e quantas mãos utiliza ao efetuar o serviço;
- **L_OR_R_Opponent:** informa se o jogador oponente é destro, ambidestro ou esquerdino e corresponde à primeira parte da variável HandOpponent;
- **Birthday:** data de nascimento do jogador principal;
- **BirthdayOpponent:** data de nascimento do jogador oponente;
- **NamesSorted:** contém um array com o nome do jogador principal e do seu oponente, eliminada após a eliminação dos jogos em espelho;
- **PlayerComb:** contém o nome do jogador principal vs o nome do jogador oponente, também por ordem alfabética. Tal como a variável anterior, foi eliminada após a retirada de jogos em espelho;
- **DifRank:** diferença em módulo entre as variáveis RankPlayer (rank do jogador principal) e RankOpponent (rank do jogador oponente), pelo que nos dá as

diferenças de ranks entre dois jogadores aquando da realização de um certo torneio em determinado ano;

- **AgePlayer:** idade do jogador principal aquando do início de certo torneio;
- **AgeOpponent:** idade do jogador oponente aquando do início de certo torneio;
- **DifAge:** diferença absoluta das idades entre AgePlayer (idade do jogador principal) e AgeOpponent (idade do jogador oponente) aquando de um certo torneio;
- **NumberWinsPlayer:** número de jogos ganhos no Brasil no total pelo jogador principal;
- **NumberWinsOpponent:** número de jogos ganhos no Brasil no total pelo jogador oponente;
- **DifNumberWins:** diferença em módulo do valor presente nas variáveis NumberWinsPlyer e NumberWinsOpponent;
- **DifHeight:** diferença em módulo da altura do jogador principal (Height) e o seu oponente (HeightOpponent);
- **NumGamesPlayerByYear:** número de jogos que o jogador principal jogou durante determinado ano;
- **NumGamesOpponentByYear:** número de jogos que o jogador oponente participou durante determinado ano;
- **DifNumGamesByYear:** módulo da diferença entre NumGamesPlayerByYear e NumGamesOpponentByYear.

Para além destas variáveis, na construção dos modelos ainda contruímos algumas variáveis dummy que serão explicadas na secção da modelação.

Estrutura do dataset

A estrutura do nosso dataset filtrado para o Brasil veio a sofrer alterações ao logo de todo o percurso, começando com 37367 linhas e 24 variáveis aquando foi filtrado e acabando por ser composto no início da fase da modelação por 19826 linhas e 42 variáveis: *PlayerName*, *RankPlayer*, *Born*, *BornCity*, *BornCountry*, *Birthday*, *AgePlayer*, *Height*, *Hand*, *L_OR_R*, *LinkPlayer*, *Tournament*, *Location*, *City*, *Country*, *Date*, *DateStart*, *DateEnd*, *Ground*, *Prize*, *GameRound*, *Oponent*, *RankOponent*, *BornOpponent*, *BornCityOpponent*, *BornCountryOpponent*, *BirthdayOpponent*, *AgeOpponent*, *HeightOpponent*, *HandOpponent*, *L_OR_R_Opponent*, *WL*, *Score*, *NumberSets*, *DifRank*, *DifAge*, *Year*, *NumberWinsPlayer*, *NumberWinsOpponent*, *DifNumberWins*,

DifHeight e *ano*. Esta diferença vem da eliminação das linhas com jogos em espelho e da criação de variáveis que consideramos relevantes e interessantes no nosso modelo.

Relativamente ao tipo de variáveis antes da fase de modelação, 23 são do tipo *object*, 10 *float*, 3 *datetime* e 6 *int*, pelo que o tipo mais frequente é *object*.

Relativamente à análise descritiva e à descoberta de outliers ou valores omissos, esta foi feita, após a eliminação de jogos em espelho e até à data tínhamos apenas 40 variáveis, faltando as variáveis *DifHeight* e *ano*. Esta foi uma estratégia adotada, pois só assim conseguimos dar uma melhor resposta aos valores omissos, mesmo assim, ainda foi necessária na secção de modelação criar novas variáveis que não vão ser explicadas nesta secção.

O número máximo de observações era de 19826, e 21 variáveis estavam nesta situação, pelo que apenas 19 tinham valores omissos e a variável com menos observações preenchidas era *BornOpponent* com apenas 10469 valores. Os valores distintos das variáveis variam de 1 em Country a 2431 em *Oponent*.

O intervalo de variação das variáveis numéricas varia muito, sendo que o desvio padrão vai de 0.5 em *NumberSets* a 211161.1 no *Prize*.

Relativamente à idade de um jogador a média é cerca de 23 anos, 25% dos jogadores tem altura inferior a 178.0 cm. O rank dos jogadores varia enormemente desde os que estão no topo da escala até aos menos conhecidos. O número de vitórias dos jogadores varia sensivelmente de 0 a 263.

Qualidade dos dados

Não foram encontrados dados duplicados na nossa base de dados, mas eliminamos os jogos em espelho, que representam 46.1% do *dataset* filtrado inicial.

Relativamente aos valores omissos, das 40 variáveis mencionadas acima, 19 tinham valores omissos, que iam desde 3 em Score até 4647 em *BornCity*.

Na nossa base de dados, através da análise descritiva também foi possível percebermos que existiam *outliers* nas variáveis *Height* e *HeightOpponent*, considerando que os valores destas variáveis iam de 0 a 510 cm.

4. Preparação dos Dados

Dataset Inicial

Com o objetivo de prever o número de sets necessários para a conclusão de um jogo de ténis profissional no Brasil, seguimos uma metodologia composta por diversos passos, desde a limpeza dos dados, criação de novas variáveis, substituição de valores omissos pelos valores reais, à visualização dos dados para perceber se existiam *outliers*, que pudessem condicionar de forma severa os resultados obtidos no nosso modelo, diminuindo a sua eficiência.

O *dataset* inicial com todos os países era composto por 1308835 linhas e 16 colunas. No entanto, apesar de o nosso foco principal ser o Brasil e do *dataset* inicial conter vários países diferentes que não correspondem ao nosso objetivo de negócio, decidimos implementar algumas mudanças neste, porque futuramente a Confederação Brasileira de Ténis podia-nos contactar de novo com o objetivo de comparar os resultados do Brasil com outro país que considerasse relevante, o que implicaria que não seria necessário repetir o processo na sua íntegra.

O *dataset* inicial continha as variáveis *_id*, *PlayerName*, *Born*, *Height*, *Hand*, *LinkPlayer*, *Tournament*, *Location*, *Date*, *Ground*, *Prize*, *GameRound*, *GameRank*, *Oponent*, *WL* e *Score*, sendo que todas eram do tipo *object*.

Alterações ao *dataset* inicial

Começamos por uniformizar a variável *Prize*, ou seja, retiramos as vírgulas presentes a cada três algarismos e retiramos as unidades monetárias, porque nem todas estavam identificadas e, assim seria mais fácil percebermos e tratarmos do dinheiro como igual, até porque ao longo dos anos a inflação variou, o que implica que não seria correto a comparação do prémio monetário, sem ter esta em conta, tal como o valor de cada moeda varia ao longo dos anos. Além disso, transformamos esta variável para o tipo *float*.

De seguida, criámos duas variáveis *DateStart* e *DateEnd* a partir da variável *Date*, sendo que a primeira corresponde ao valor antes do símbolo '-' e a segunda ao valor depois deste símbolo. É de salientar que nos casos em que apenas existia um valor consideramos que a data de fim do torneio era igual à data de início, o que fez com que deixássemos de ter 12390 valores omissos na variável *DateEnd*.

Também reparamos que existiam outras variáveis com valores omissos para além da *DateEnd*, como é o caso de *Height* com 326177 linhas omissas, *Prize* com 19136 e *GameRank* com 119055, sendo que apenas tratamos da imputação dos valores omissos das restantes variáveis, depois de termos filtrado os dados para o Brasil, porque ainda não era relevante fazê-lo numa fase tão inicial do trabalho, considerando que o nosso cliente não pretende ter um modelo preditivo para todos os países e o número de valores omissos para o Brasil seria certamente bastante inferior.

Decidimos mudar o nome da variável *GameRank* para *RankOpponent*, uma vez que o nome inicial era ambíguo, considerando o verdadeiro significado desta variável, se prende não com o *rank* do jogo em si, mas com o *rank* do jogador oponente, ou seja, o jogador cujo nome se encontra na coluna *Oponent*.

Criamos a função *get_rank_by_year*, para que pudéssemos saber o rank do jogador principal em determinado ano, sendo que o ranking dos jogadores que competem em competições ATP muda todas as semanas, geralmente à segunda-feira, consoante o desempenho dos jogadores nas últimas 52 semanas e os pontos que levam ao cálculo do ranking são atribuídos consoante o tipo de torneio e a sua performance neste. Esta função procura pelo campo *DateStart* se já existe um rank para esse jogador enquanto oponente, ou seja, procura nas colunas *Oponent* e *RankOpponent*. Em seguida, cria um dicionário para cada ano com o nome do jogador e o ranking correspondente naquele ano. A função *get_rank* procura o valor do ranking do jogador principal (*PlayerName*) no dicionário e cria a coluna *RankPlayer* com o *rank* do jogador principal. A nova variável ficou com 77272 valores omissos, ainda que desses apenas 8135 fossem de jogadores distintos.

Decidimos uniformizar a forma de apresentação da variável *Score*, considerando que cada par, por vezes, se encontrava separado por vírgulas e existiam espaços extras entre pares, que não são necessários e foram eliminados.

Com base na variável *Score*, criamos a variável *NumberSets*, que conta o número de pares presentes na variável Score, sendo que mais à frente na parte de modelação decidimos não considerar os valores "RET" como um *set* realizado, pois esta é uma indicação de que um jogador se retirou do jogo, por exemplo, por motivos de doença ou lesões, o que leva a que o outro jogador seja automaticamente declarado vencedor. Assim, optamos por não o considerar como um *set*, seguindo o raciocínio de que este não aconteceu na realidade e eliminamos as linhas com RET na fase de modelação.

Utilizamos a variável *_id* do nosso *dataset* para garantir que não existiam duplicados, tal como referido e criamos a variável *_id_str* a partir desta, na qual convertemos o dicionário (*_id*) em *strings* (*_id_str*). Através desta, comprovamos que não existiam linhas duplicadas e eliminamos em seguida, as variáveis *_id* e *_id_str*.

A partir da coluna *Location*, criamos a variável *City*, que contém informações acerca da cidade de realização do torneio e esta foi conseguida, ao extrairmos a primeira parte da variável *Location*, ou seja a informação que se encontrava à esquerda da vírgula.

De seguida, fomos à procura de um *dataset* que contivesse cidades e o país correspondente, para podermos uniformizar e completar eventuais valores omissos presentes na variável *Location* que é composta pela cidade e país correspondente. Assim, importamos o csv worldcities.csv, composto por 44691 linhas e 11 colunas. Como reparamos que no *dataset* original o mapeamento não estava sempre correto, decidimos verificar se existiam valores duplicados na coluna *city_ascii* e na coluna *country* do novo csv, ou se existem cidades com nomes diferentes que correspondem ao mesmo local, o que estava a provocar uma sobreposição do mapeamento das cidades com o mesmo nome em países diferentes, pelo que eliminamos os duplicados para que o mapeamento ficasse correto.

Deste modo, criamos a variável *Country*, a partir do csv importado e da variável *City*, criada no *dataset* a partir do campo *Location*. A nova variável apresenta 441437 valores omissos.

Percebemos que esta situação se verificava quando as variáveis *Location* e *Country* tinham apenas o nome do país e considerando que o cruzamento entre *datasets* era feito por cidades, não se conseguia completar o valor de *Country*. Deste modo, decidimos que o valor que iria ficar em *Country* seria o que estivesse na *City* correspondente, o que fez com que deixássemos de ter valores omissos.

De seguida, criamos a variável *BornCity* a partir da informação que se encontrava à esquerda da vírgula da variável *Born* e a variável *BornCountry* com base no mapeamento entre cada país e cidade, ou seja, voltamos a usar o *dataset worldcities.csv*. Ao início tínhamos 466332 linhas omissas, mas após substituirmos com os valores presentes em *BornCity* da linha correspondente, ficamos com 320752 valores omissos.

Para ainda diminuir mais este valor, fomos à procura de uma base de dados que nos ajudasse e importamos o csv *player_overviews_unindexed_csv.csv* com 10912 linhas e 20 colunas. Neste *dataset* criamos a coluna *Name* que junta as colunas *first_name* e *last_name* para ficar no mesmo formato que a coluna *PlayerName* do *dataset* inicial com o qual estamos a trabalhar.

De forma a cruzar a informação do *dataset player_overviews_unindexed_csv.csv* com o *dataset* das cidades e países (*worldcities.csv*), decidimos utilizar o código *iso3* para cruzar a informação entre as bases de dados.

Criámos a função *fill_missing_born_country* que cria um dicionário que mapeia os nomes de jogadores e os respetivos países de nascimento e preenche os valores ausentes em *BornCountry*, usando este mapeamento. No entanto ainda ficamos com 192459 valores omissos.

De seguida, criamos a variável *L_OR_R*, que contém a primeira parte da variável *Hand*, ou seja, contém a informação à esquerda da vírgula e não aparenta ter valores omissos.

Deste modo, o *dataset* inicial passa a ser composto pelo mesmo número de linhas - 1308835 - e por 24 colunas: *PlayerName*, *RankPlayer*, *Born*, *BornCity*, *BornCountry*, *Height*, *Hand*, *L_OR_R*, *LinkPlayer*, *Tournament*, *Location*, *City*, *Country*, *Date*, *DateStart*, *DateEnd*, *Ground*, *Prize*, *GameRound*, *Oponent*, *RankOpponent*, *WL*, *Score*, *NumberSets*. Relativamente à existência de valores omissos, temos as variáveis *RankPlayer* com 77272, *BornCity* com 320752, *BornCountry* com 192459, *Height* com 326177, *Prize* com 19136, *RankOpponent* com 119055 e *Score* com 41.

Alterações ao *dataset* ATP Brasil

De seguida, focamo-nos no nosso país de estudo: o Brasil.

Começamos por avaliar as diferentes formas que o Brasil poderia estar escrito para conseguirmos fazer um estudo correto deste país, de outro modo, poderíamos estar a descartar observações relevantes. Desta forma, procuramos na coluna *Country* todos os nomes começados por 'br', sendo que a procura não foi *case sensitive*. Dos países que nos apareceram, percebemos que os únicos que efetivamente nos interessavam eram "Brazil" e "Brazi". Assim, decidimos uniformizar a escrita de "Brasil", ou seja, transformamos "Brazi" em "Brazil", considerando que a última é a forma de escrita correta em inglês.

Considerando algumas metas/instruções que nos foram dadas pelo nosso cliente, elaboramos algumas das variáveis criadas acima também para o jogador oponente, ou seja, para o jogador presente na variável *Oponent*. Para tornar o processo mais eficiente, considerando que vamos cruzar com a informação já existente para o jogador - *PlayerName* -, optámos pela criação destas apenas no *dataset* filtrado para o Brasil.

Começamos por filtrar no *dataset* do Brasil, as linhas com os nomes dos jogadores (*PlayerName*) distintos, o que torna o cruzamento da informação mais eficiente na criação das variáveis para o oponente (*Oponent*).

A criação da coluna *BornOpponent* foi feita procurando o valor correspondente de *Born*, quando este jogador se encontrava na coluna *PlayerName*.

O processo análogo foi adotado na criação da coluna *BornCityOpponent* e *BornCountryOpponent*, ou seja, procurou-se se o jogador oponente era o jogador principal noutra linha e se tinha um valor em *BornCity* para a primeira e se tinha um valor em *BornCountry* para a última. Um processo em tudo similar, foi adotado para a variável *HeightOpponent*, *HandOpponent* e *L_OR_R_Opponent*, onde se foi procurar o valor correspondente em *Height*, *Hand* e *L_OR_R*, respetivamente.

Com vista à criação das variáveis *BirthDate* e *BirthDateOpponent* procedemos à procura de um *dataset* que contivesse o nome dos jogadores e o seu dia de aniversário e para isso foi importado o *dataset Player.csv* com 53666 linhas e 5 colunas. De seguida, criou-se a coluna *BirthDate* e *BirthDateOpponent* vazias no *dataset* inicial filtrado para o Brasil. Para a primeira, criámos um dicionário com os valores da variável *birthday* presente no *dataset* importado mapeados para cada jogador. Com base no mapa de aniversários criados, foram preenchidos os valores "NaN", para os valores correspondentes aos aniversários dos jogadores presentes na coluna *PlayerName*. A partir desse mesmo mapeamento preenchemos os valores "NaN" presentes em *BirthDateOpponent* para os aniversários dos jogadores presentes em *Oponent*.

Jogos Espelhados

Algo que já vínhamos a reparar era a existência de jogos em espelho, o que que tinha de ser tratado, pois condicionaria de grande forma os resultados obtidos, considerando que estes estavam repetidos duas vezes, apenas mudando as variáveis *PlayerName* e *Oponent* e variáveis associadas. Para eliminarmos estes casos, criamos duas variáveis:

NamesSorted, que ordena os nomes dos jogadores presentes em *PlayerName* e *Oponent* por ordem alfabética e *PlayerComb*, que cria uma coluna com a combinação dos nomes, mantendo a ordenação alfabética. Assim, decidimos filtrar os jogos únicos considerando o torneio -*Tournament*-, a data -*Date*- e os jogadores que participaram na partida – *PlayerComb*. Deste modo, passámos de 37367 linhas, para apenas 20138, ou seja, os jogos em espelho representavam 46.1% do *dataset* filtrado. Após a não existência deste problema, foram eliminadas as colunas referentes às variáveis *NamesSorted* e *PlayerComb*, pois elas só foram criadas com o objetivo de auxiliar a eliminação dos jogos em espelho.

De seguida, procedemos à verificação da estrutura e qualidade dos dados, de forma resumida, na qual reparamos de imediato na existência de valores omissos.

Valores Omissos – Web Scrapping

Devido à existência de valores omissos em 15 variáveis, decidimos de seguida começar por tratar desses casos.

Procedemos ao Web Scrapping nos websites ATP Tour e Tennis Explorer a fim de preenchermos os valores omissos de algumas variáveis. Para realizar esta técnica, foram utilizadas as bibliotecas "requests_html" e "BeautifulSoup". A primeira permite fazer requisições HTTP e extrair informações de páginas da web, enquanto a segunda é uma biblioteca para análise e extração de dados HTML.

No site ATP, realizámos Web Scrapping para as variáveis *Height*, *Birthday*, *Location* e consequentemente *City* e *Country*. Este foi realizado usando como ponto de entrada o link de cada jogador já disponível na base de dados na variável *LinkPlayer*.

Contudo, foi manifestamente insuficiente, pelo que recorremos ao site do *Tennis Explorer* de onde extraímos a informação. No entanto, verificámos que, em alguns casos, os links dos jogadores não seguiam o mesmo formato, pelo que houve a necessidade de criar para os jogadores em falta um dicionário entre o seu nome e o respetivo *link*.

Para este site, além de completarmos alguns casos de omissos para as variáveis anteriormente mencionadas quando realizamos o Web Scrapping no site ATP, completamos as variáveis *RankPlayer* e *RankOpponent*, relacionadas com os *ranks* dos jogadores.

No caso do *rank* do jogador (*RankPlayer*), do *rank* do oponente (*RankOponent*) e da data de nascimento do jogador (*Birthday*) houve a necessidade de pesquisar manualmente a fim de preencher alguns valores em falta, sendo que no caso do *RankPlayer* conseguimos, depois deste processo, não ter qualquer valor em falta.

Considerando que já tínhamos feito Web Scrapping para as variáveis relativas aos jogadores principais, aproveitamos para cruzar a informação com os jogadores oponentes, o que resultou em termos menos linhas com valores omissos, tornando o processo mais eficiente.

As linhas em que o *Oponent* apresentava o valor "bye" foram eliminadas, pois indica que não houve jogo, ou seja, o jogador presente em *PlayerName*, avançou automaticamente para a próxima ronda por falta de oponente. Esta oportunidade é dada maioritariamente a jogadores mais conceituados e acontece quando o número de jogadores não é par. É de notar que estes valores "bye" também adicionavam vários omissos ao *dataset*, considerando que todas as variáveis associadas ao oponente e ao jogo não puderam ser preenchidas (uma vez que não se realizou nenhuma partida).

Com vista já às variáveis necessárias para a criação do modelo, procedeu-se à criação da variável *DifRank*, que nos dá a diferença em valor absoluto dos *ranks* entre o jogador principal -*RankPlayer*- e o jogador oponente – *RankOpponent*.

Com o mesmo objetivo, criámos a variável *DifAge* e as variáveis *AgePlayer* e *AgeOpponent* necessárias para a criação da primeira. Começámos por criar a coluna *AgePlayer* com os valores todos a "NaN". Transformámos a variável *Birthday* no tipo *datetime* e subtraímos a data de começo do torneio -*DateStart*- com a data de nascimento -*Birthday*. De seguida, foi feito o processo análogo para *AgeOpponent*, só que neste caso a diferença foi realizada entre as variáveis *DateStart* e *BirthdayOpponent*. Por fim, elaborámos a coluna *DifAge* inicialmente com os valores "NaN". A diferença de idades foi obtida em módulo e através da subtração das variáveis *AgePlayer* e *AgeOpponent*.

Também criamos a função *count_wins* que conta o número de vitórias do jogador principal (PlayerName) e jogador oponente (Opponent) em torneios realizados no Brasil, o que resultou na criação das variáveis *NumberWinsPlaye*r e *NumberWinsOpponent*, respetivamente.

Por fim, a coluna *DifNumberWins* foi criada, através do módulo da diferença entre *NumberWinsPlayer* e *NumberWinsOpponent*.

Análise Descritiva e Qualidade dos Dados

Após a criação das variáveis descritas acima, decidimos averiguar a qualidade dos dados, ou seja, a existência de *outliers* e valores omissos.

Relativamente à existência de valores omissos, estes existiam em 19 das 40 variáveis, sendo que em algumas os valores chegavam à casa dos milhares.[1]

A análise descritiva realizada foi dividida entre as variáveis categóricas e numéricas. Relativamente às primeiras, conseguimos informação acerca do número de linhas preenchidas, o que nos permite saber quantas linhas têm valores omissos, o número de valores únicos, a moda e a frequência deste valor. Deste modo, das 22 variáveis, apenas 11 estão completas, o número de valores únicos varia de 1 em *Country* a 2431 em *Oponent*. Referente às variáveis numéricas estas são 14 e temos informação acerca do número de observações, da média, do desvio padrão, do valor mínimo e máximo, da média e do quartil 25, 50 e 75. Destas apenas 6 variáveis têm valores omissos. A média varia de 2.3 em *NumberSets* a 70127.6 em *Prize*, pelo que temos variáveis com grandezas bastante diferentes. Em algumas variáveis nota-se a existência de valores impossíveis no contexto, como por exemplo, o caso da variável *Height* que varia de 0 a 510 cm.

Ao analisarmos os valores únicos percebemos que estes valores variavam muito, ou seja, iam de 1 em *Country* a 2431 em *Oponent*.

Ao analisarmos os valores omissos [2]percebemos que a percentagem de omissos chega a quase 50% em três variáveis, nomeadamente *BornOpponent*, *BornCityOpponent* e *HeightOpponent*.

Considerando o elevado número de valores omissos e a existência de valores impossíveis, fomos investigar essa parte e tentamos adotar uma estratégia que permitisse que estes deixassem de existir.

Começamos por contar o número de linhas distintas, em que o torneio não contém informação acerca do prémio recebido, o que resultou no número 37 e percebemos que estes valores aconteciam principalmente em competições entre países, ou seja,

[1] Verificar anexo 1
[2] Verificar Anexo 2

pertencentes à Davis Cup. Criamos a função *preencher_missings_por_torneio* que preenche o Prize para um dado torneio se este já existir noutra linha, no entanto não houve diminuição da quantidade de valores omissos, pelo que decidimos pesquisar um pouco mais sobre estas competições e percebemos que nesta os jogadores apenas recebiam uma medalha e não existia um prémio monetário associado ao torneio, pois este era atribuído pelo país do respetivo jogador. Desta forma, decidimos imputar por zero os valores omissos, considerando que este é o valor mais admissível considerando que efetivamente o torneio em si não tem prémio associado.

De seguida, decidimos tratar da variável *Height* que tinha 4358 valores omissos, mas destes apenas 448 correspondiam a jogadores diferentes. Criamos a função *fill_missing_height* para verificar se a altura desse jogador já se encontrava presente noutra linha, mas esta situação não se verificou. Decidimos tratar os valores dos *outliers*, para percebermos melhor quais os seus valores criamos um *boxplot* [3], no qual nos foi possível verificar a existência de valores acima de 500 e abaixo de 25 cm. O jogador com 510 cm era Carlos Di Laura e a sua verdadeira altura é de 175 cm, como foi possível encontrar na internet e optamos por fazer uma substituição de um valor por outro. Relativamente aos 30 jogadores com altura igual a 0 cm, optamos igualmente por procurar o seu valor real na internet, mas em muitos casos a busca foi infrutífera, o que levou a que preenchêssemos os valores que não conseguimos encontrar com alturas entre 170 e 195 cm, considerando que a maioria dos jogadores de ténis se encontram entre estas alturas. Verificámos a existência de um jogador com menos de 25 cm, chamado Grant Stafford e aplicámos um método análogo ao aplicado para Carlos Di Laura. O gráfico após a substituição dos valores, pode ser encontrado no Anexo 4[4].

Relativamente aos valores omissos, criamos a função *preencher_alturas_faltantes* para ver a altura desse jogador já existia noutra linha, situação que não se verificou.

Um processo em tudo semelhante foi realizado para a variável *HeightOpponent*. É de notar que não imputamos valores para as variáveis relacionadas com a altura do jogador, pois faltavam um número considerável de observações, logo se imputássemos tantos valores, estaríamos a artificializar o modelo em demasia, logo com outros dados, a nossa proposta poderia não funcionar tão bem.

[3] Ver Anexo 3
[4] Ver Anexo 4

Nesta fase, criamos a variável *DifHeight* que nos dá o valor referente à diferença de alturas entre *Height* e *HeightOpponent*.

Como reparamos que a variável *Score* tinha alguns valores omissos, os que conseguimos encontrar na internet foram substituídos no *dataset*.

De seguida, realizamos alguns gráficos de visualização, para as variáveis numéricas e categóricas[5], onde foi possível ver a existência de valores omissos e os valores admissíveis para cada variável. Considerando os jogos de 1998 a 2022, o ano com mais jogos foi em 2018[6].

Assim, antes da Modelação o nosso dataset tinha 19826 linhas e 42 colunas, do tipo object, float, datetime e int e no pior dos casos 10303 valores omissos na variável *DifHeight*.

Por fim, na secção de Modelação, ainda criamos a variável referente à diferença absoluta de jogos por época entre o jogador principal e oponente (*DifNumGamesByYear*). O valor resulta da diferença das variáveis *NumGamesOpponentByYear* e *NumGamesPlayerByYear*, que conta o número de jogos que certo jogador presente na variável *Oponent* e *PlayerName*, respetivamente jogou num certo ano.

Modelação

Nesta fase, desenvolvemos diversos modelos de classificação, com diversas variáveis numéricas e categóricas e combinações das mesmas e diferentes algoritmos para tentarmos chegar ao melhor modelo, com os melhores resultados e que pudesse ser aplicado noutro conjunto de dados e continuar a dar bons resultados.

Antes de começarmos a explicar os procedimentos utilizados para construir os diferentes modelos, vamos explicar os diversos algoritmos utilizados, pois estes vão auxiliar a compreensão de todas as etapas realizadas. Deste modo, os algoritmos escolhidos foram:

- **Random Forest:** algoritmo que combina árvores de decisão e o método de *bagging* (*boostrap aggregating*), que é utilizado para melhorar a precisão e robustez dos modelos e consiste na seleção aleatória e com reposição de diversas amostras criadas a partir do conjunto de treino. O modelo preditivo é treinado para

[5] Ver Anexo 5
[6] Ver Anexo 6

cada uma dessas amostras. Neste projeto, utilizamos 5 e 10 *folds* para treinar o modelo. Neste caso, construímos árvores a partir de uma amostra aleatória do conjunto de dados original, ou seja, cada árvore é treinada num subconjunto diferente de dados.

- **Decision Tree:** as árvores de decisão são construídas a partir do conjunto de treino e devem conseguir mapear as características de certa classe. Em cada nó, existe a seleção de uma característica, sendo um processo recursivo. Também as utilizamos para fazer a *feature importance* das variáveis utilizadas em cada modelo, sendo uma medida da importância relativa dessa variável na previsão.
- **Gradient Boosting:** pertence aos métodos de *boosting*, que combinam modelos mais fracos - estimadores base - para criarem um modelo preditivo mais eficaz, ou seja, é um treino iterativo, onde cada novo modelo criado é construído para corrigir as falhas dos modelos anteriores. O Gradient Boosting utiliza o gradiente descendente para minimizar a função de perda durante o treino. Em cada iteração, procura a direção no espaço de características que conduzirá à redução do erro de previsão.
- **AdaBoost:** é um método de *boosting* que pretende melhorar a previsão, em problemas de classificação binária, atribuindo pesos diferentes às amostras durante o treino, sendo que as que foram mal classificadas têm um peso maior para que consigam ser corrigidas. Este algoritmo consegue lidar com a não linearidade e complexidade dos dados, sendo menos propenso a *overfitting*, mas pode ser sensível a *outliers* e ruído nos dados de treino
- **XGBoost:** é um algoritmo baseado no Gradient Boosting, mas com algumas melhorias em comparação com este. Tem termos de penalização na função de perda (medida que quantifica a discrepância entre os valores previstos e os valores reais) durante o treino. Cada modelo é treinado com o objetivo de minimizar a função de perda, usando o gradiente descendente para atualizar os pesos das amostras. Consegue lidar com dados complexos e permite o ajuste dos hiperparâmetros para obter o melhor desempenho.

Agora, relativamente aos passos realizados, após a eliminação dos RET no Score, ficamos com 19208 linhas pois não faz sentido considerar jogos em que um jogador abandona ou desiste da partida. Também decidimos eliminar os valores omissos, pois nem todos os

modelos utilizados conseguem fazer previsões quando os dados não estão completamente preenchidos.

Começamos por dividir os dados em dois conjuntos: um continha os jogos à melhor de 3, ou seja, jogos cujo número de sets vai de 1 a 3 e outro continha os jogos à melhor de 5, ou seja, jogos com 4 ou 5 sets, característicos dos torneios Grand Slam.

De seguida, realizamos a matriz de correlação utilizando os dados de treino dos jogos à melhor de 3, apenas com as variáveis numéricas (*DifHeight, DifRank, Prize, DifNumberWins*), usando a correlação de Pearson[7], onde a correlação mais alta era de 0.19 entre *DifRank* e *DifNumberWins* e a mais baixa entre *DifHeight* e *DifRank*. Utilizamos V de Cramer com as variáveis categóricas (*NumberSets, L_OR_R_Left-Handed,, L_OR_R_Right-Handed, L_OR_R_Opponent_Left-Handed, L_OR_R_Opponent_Right-Handed, Ground_Clay e Ground_Hard*)[8], onde as correlações foram muito baixas e abaixo de 0.01. É de notar que decidimos transformar as variáveis *L_OR_R, L_OR_R_Opponent* e *Ground* em variáveis dummy. Este já foi um movimento estratégico adotado, considerando as variáveis categórica criadas ou iniciais que tínhamos no *dataset* e que pensamos serem relevantes na criação dos modelos intermédios que nos irão levar ao modelo final eficaz, preciso e exato.

Também verificamos a multicolinearidade, pois se existissem variáveis com valores de VIF acima de 5, podíamos eliminar uma dessas variáveis, permitindo a simplificação do modelo, mas como o maior valor de VIF encontrado foi de cerca de 3.2, mantivemos todas as variáveis na nossa base de dados.

De seguida, realizamos dois modelos iniciais, um apenas com jogos à melhor de 3 e outro com jogos à melhor de 5. Optamos por não realizar um modelo que juntasse torneios com 1 a 5 *sets*, pois os jogos à melhor de 5, são extremamente raros na nossa base de dados, não sendo representativos desta, logo têm pouco poder preditivo.

Vamos começar por explicar o modelo à melhor de 5, para jogos com 4 ou 5 *sets*, pois este tem pouca representatividade no nosso *dataset* e os modelos à melhor de 3 são mais complexos e passaram por várias transformações. Para jogos com 4 ou 5 *sets* temos apenas 46 linhas na base de dados referentes ao Brasil, o que representa um valor muito pequeno considerando a grande dimensão da base de dados. Mesmo assim, decidimos

[7] Ver Anexo 7
[8] VerAnexo 8

avançar com o objetivo de tentar obter algum insight sobre estes dados. Assim, dividimos o conjunto de treino e teste em 70-30%, respetivamente, o que resultou em 32 linhas no treino e 14 linhas no teste.

Para este modelo, decidimos usar 4 variáveis, nomeadamente a *DifNumberWins, DifRank, DifAge e DifHeight.* Acreditamos na relevância de *DifNumberWins*, pois um jogador que tenha ganho mais jogos, à partida tem mais experiência, mais técnica, estratégia e *skills* o que pode levar a um jogo com menos *sets*. A *DifRank*, na medida em que, jogadores mais conceituados e com rankings mais perto do topo da escala têm mais experiência e *skills*, pelo que pelo menos em etapas iniciais das competições onde jogam com jogadores menos conceituados, devem conseguir acabar o jogo mais depressa. A *DifAge*, pois a idade condiciona a condição física, força, agilidade e a propensão para lesões, sendo a sua recuperação dificultada. Por fim, a *DifHeight*, pois se a diferença de altura for muito grande, pode haver um desequilíbrio no jogo, pois o mais alto consegue cobrir mais terreno, mas pode ser mais lento nas movimentações que o jogador mais baixo.

Na Figura 1, podemos verificar que a variável que contribui mais para a previsão é *DifAge* com pouco mais de 0.35.

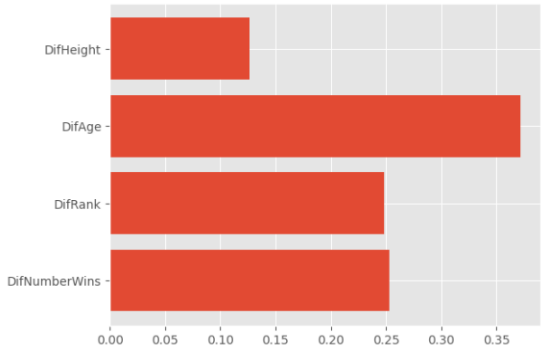


Figura 1-Feature Importance para o modelo com 4 ou 5 sets

Relativamente a algoritmos, exploramos o Random Forest com 5 *folds*, a Decision Tree e a Feature Importance e o Gradient Boosting com 20 iterações e a curva ROC da última (Tabela 1).

Modelos	Accuracy	Precision	Recall	F1-Score	Support	AUC
Random Forest - 5 folds	0.75	-	-	-	-	-
Decision Tree	0.21	0.27 0.00	0.50 0.00	0.35 0.00	6 8	-
Gradient Boosting	0.29	0.30 0.25	0.50 0.12	0.37 0.17	6 8	0.30

Tabela 1-Resultados modelo com 4 ou 5 sets

Através da Tabela 1, percebemos que os valores alcançados, são muito pobres e reparamos de imediato no reduzido número de observações do teste.

Pela última razão, decidimos não aprofundar o estudo destes casos, pois seria difícil de generalizar para um caso real com tão poucos dados e seguindo as indicações dadas pelo nosso cliente focarmo-nos nos casos à melhor de 3, porque são os mais frequentes e por isso têm mais interesse considerando o objetivo de negócio dos nossos clientes.

Assim, relativamente aos modelos à melhor de 3, ou seja, com 1, 2 ou 3 *sets* fizemos vários modelos, sempre com o objetivo de os melhorar progressivamente, porque pensamos que estes são mais interessantes de explorar principalmente o caso com 2 ou 3 *sets*, por razões que vamos explorar de seguida. Deste modo, vamos fazer uma breve explicação dos modelos intermédios, passos realizados e que conclusões retiramos em cada um deles até chegarmos ao melhor modelo e algoritmo que lhe dá os melhores resultados que vamos apresentar em mais detalhe. É de notar que vamos atribuir um número a cada modelo, o que vai tornar a sua explicação e avaliação de resultados mais fácil.

Modelo 1: No primeiro modelo que realizamos, consideramos jogos com 1, 2 ou 3 sets, e uma divisão 70 - 30%, o que resultou em 5852 linhas no treino e 2509 linhas no teste, respetivamente. É de notar que decidimos realizar esta divisão, para prevenir o *overfitting* e ajudar aquando do ajuste dos hiperparâmetros e na validação do modelo. Para o Modelo 1 decidimos utilizar apenas as variáveis *DifNumberWins*, *DifRank*, *DifAge* e *DifHeight*, sendo que a mais relevante é *DifRank* com cerca de 0.40 de importância relativa (Figura 2).

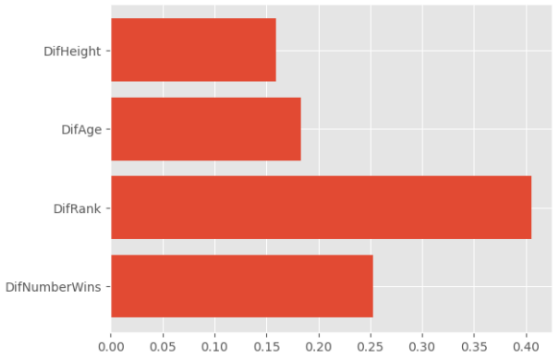


Figura 2-Feature Importance Modelo 1

Realizamos Random Forest com 5 *folds*, Decision Tree e feature importance, Gradient Boosting e a curva ROC desta (Tabela 2).

Modelos	Accuracy	Precision	Recall	F1-Score	Support	AUC
Random Forest - 5 folds	0.61	-	-	-	-	-
Decision Tree	0.55	0.0 0.66 0.34	0.00 0.65 0.36	0.00 0.65 0.35	9 1648 852	-
Gradient Boosting	0.66	0.0 0.66 0.00	0.00 1.00 0.00	0.00 0.79 0.00	9 1648 852	0.53

Tabela 2-Resultados Modelo 1

Vemos que temos 3 classes (1, 2, 3), os resultados em termos de *Accuracy* para primeiro modelo são aceitáveis, mas a previsão para *sets* de 1 é algo pobre.

Modelo 2: Ao verificarmos que a grande maioria dos *sets*, apenas tinha 2 ou 3 sets, decidimos experimentar, as mesmas variáveis, só que apenas para jogos com 2 ou 3 sets. É de notar que a relevância das variáveis no Modelo 1 e 2 é igual (Figura 3).

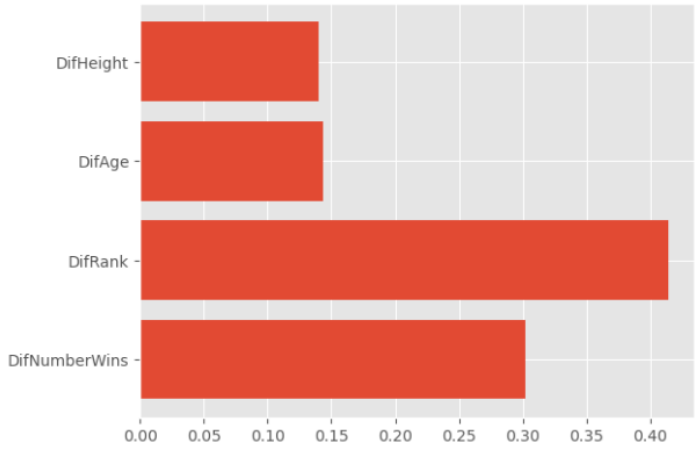


Figura 3-Feature Importance Modelo 2

Neste caso, a divisão 70-30% resultou em 5835 linhas no treino e 2502 linhas no teste. Adotamos os mesmos algoritmos, mas decidimos experimentar o Random Forest com 5 e 10 *folds*, para treinar o modelo, pois pensamos que o modelo treinado e ajustado mais vezes, pode resultar numa estimativa mais robusta acerca do desempenho do modelo. Além disso, também tentamos realizar o Gradient Boosting com 20 e 40 iterações para tentarmos obter um melhor ajuste nos hiperparâmetros e consequentemente melhores resultados; decidimos experimentar com o AdaBoost e XGBoost e criamos a curva ROC para todos os algoritmos testados. É de notar que como os resultados do Gradient Boosting com 20 e 40 iterações são iguais, apenas apresentamos os resultados do Gradient Boosting com 20 iterações (Tabela 3), o que prova que com 20 iterações, o modelo consegue ajustar da melhor forma os hiperparâmetros.

Modelos	Accuracy	Precision	Recall	F1-Score	Support	AUC
Random Forest - 5 folds	0.60	-	-	-	-	0.51
Random Forest - 10 folds	0.61	-	-	-	-	0.50
Decision Tree	0.55	0.66 0.35	0.64 0.37	0.65 0.36	1639 863	0.50
Gradient Boosting	0.66	0.66 0.60	1.00 0.00	0.79 0.01	1639 863	0.53
AdaBoost	0.65	0.66 0.40	1.00 0.00	0.79 0.01	1639 863	0.53
XGBoost	0.61	0.66 0.35	0.85 0.15	0.74 0.21	1639 863	0.50

Tabela 3-Resultados Modelo 2

Relativamente, ao Random Forest com 5 e 10 *folds*, os resultados são bastante semelhantes. Em termos de *Accuracy*, o melhor algoritmo é o Gradient Boosting, a nível do AUC, os melhores valores de 0.53 são alcançados no Gradient Boosting e AdaBoost, no entanto estes modelos, tendem a prever a moda, como se pode verificar no valor do Recall. Deste modo, consideramos que o melhor modelo é o XGBoost, pois escolher um dos outros, em casos que tenhamos dados, cujas classes estão desequilibradas iria piorar ainda mais a tendência já verificada.

É de notar que como o Random Forest é um algoritmo algo fraco, apenas medimos a *Accuracy* e o AUC, também nos modelos daqui em diante. Este nunca foi considerado um bom modelo, devido aos valores bastante baixos de AUC (prevê como um classificador aleatório), que se verificaram e devido à sua simplicidade enquanto modelo, que poderia não resultar tão bem em dados mais complexos. Além disso, a partir daqui a

metodologia e os algoritmos usados são os mesmos, só mudamos as variáveis, para tentar melhorar os resultados dos modelos.

Modelo 3: Criamos um modelo, apenas com as variáveis numéricas, como preditores, ou seja, as variáveis DifNumberWins, DifRank, DifAge, DifHeight e Prize (Figura 4). Decidimos adicionar a variável Prize enquanto tentativa de chegarmos a melhores resultados e seguindo o raciocínio de que quanto maior o prémio, mais sets ocorreriam, pois os jogadores estariam mais empenhados em ganhar.

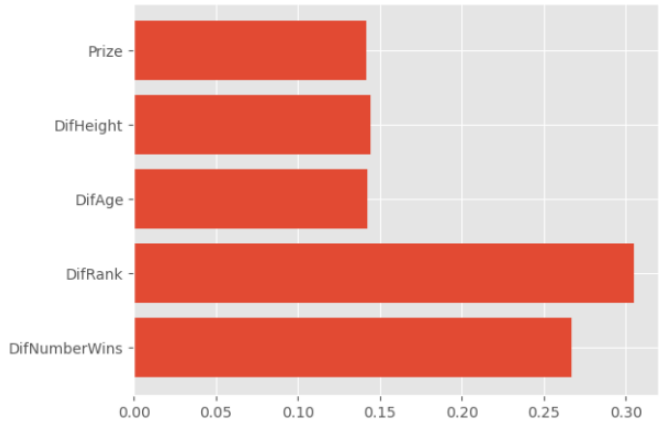


Figura 4-Feature Importance Modelo 3

Como com este modelo, o desempenho do modelo Gradient Boosting com 20 e 40 iterações voltou a ser o mesmo, só apresentamos na Tabela 4 os resultados com 20 iterações e decidimos, nos restantes modelos, apenas experimentar este algoritmo com 20 iterações, pois este número de iterações aparenta ser suficiente para ajustar os hiperparâmetros.

Modelos	Accuracy	Precision	Recall	F1-Score	Support	AUC
Random Forest - 5 folds	0.62	-	-	-	-	0.51
Random Forest - 10 folds	0.62	-	-	-	-	0.52
Decision Tree	0.56	0.66 0.35	0.67 0.34	0.66 0.34	1651 851	0.50
Gradient Boosting	0.66	0.66 0.35	0.99 0.01	0.79 0.01	1651 851	0.53
AdaBoost	0.66	0.66 0.45	0.99 0.02	0.79 0.04	1651 851	0.53
XGBoost	0.61	0.66 0.34	0.84 0.16	0.74 0.21	1651 851	0.52

Tabela 4-Resultados Modelo 3

Mais uma vez, os resultados do Random Forest com 5 e 10 *folds* foram muito parecidos. Os resultados do Gradient Boosting e AdaBoost são muito parecidos, no entanto

continuam a prever muito a moda. Dos que não prevêem a moda, o XGBoost é o com melhores resultados.

Modelo 4: De seguida, decidimos colocar algumas variáveis dummy que resultam das variáveis *L_OR_R* e *L_OR_R_Opponent*, pois a mão que utilizam pode influenciar os resultados do jogo, principalmente se estivermos a considerar de atletas que jogam com mãos diferentes, pelo que no Modelo 10 vamos explorar esta possibilidade, neste exploramos apenas as mãos utilizadas e se estas influenciam. O Modelo 4 tem as variáveis *DifNumberWins, DifRank, DifAge, DifHeight, L_OR_R_Left-Handed, L_OR_R_Right-Handed, L_OR_R_Opponent_Left-Handed* e *L_OR_R_Opponent_Right-Handed* (Figura 5).

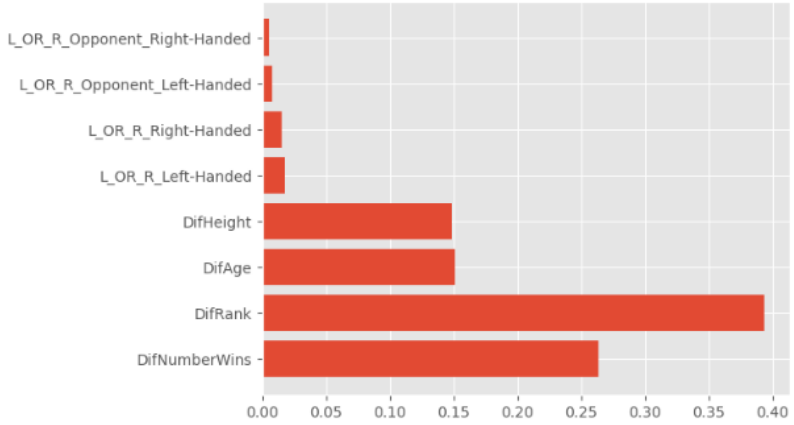


Figura 5-Feature Importance Modelo 4

As variáveis dummy contribuem com menos de 0.05 para o modelo. Na Tabela 5 temos os resultados obtidos.

Modelos	Accuracy	Precision	Recall	F1-Score	Support	AUC
Random Forest - 5 folds	0.61	-	-	-	-	0.51
Random Forest - 10 folds	0.61	-	-	-	-	0.51
Decision Tree	0.54	0.65 0.34	0.64 0.34	0.65 0.34	1638 864	0.49
Gradient Boosting	0.65	0.65 0.00	1.00 0.00	0.79 0.00	1638 864	0.53
AdaBoost	0.66	0.66 0.57	1.00 0.00	0.79 0.01	1638 864	0.53
XGBoost	0.60	0.65 0.31	0.84 0.13	0.73 0.19	1638 864	0.51

Tabela 6-Resultados Modelo 4

A melhor *Accuracy* e AUC foram obtidas no AdaBoost, no entanto este, apenas prevê a moda, pelo que o XGBoost tem de ser considerado o melhor, porque é sensível aos sets de 3.

Modelo 5: Posteriormente, colocamos também o *Prize* mais uma vez, ficando com as *variáveis DifNumberWins, DifRank, DifAge, DifHeight, Prize, L_OR_R_Left-Handed, L_OR_R_Right-Handed, L_OR_R_Opponent_Left-Handed* e *L_OR_R_Opponent_Right-Handed*. Segundo a Figura 6, a mais relevante é *DifRank* e as menos relevantes são as variáveis dummy.

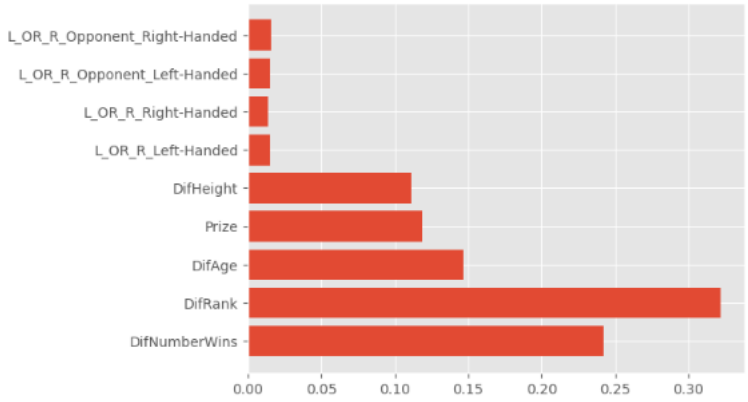


Figura 6- Feature Importance Modelo 5

Na Tabela 6, temos os resultados. As conclusões retiradas anteriormente continuam a ser válidas, ou seja, o AdaBoost e o GradientBoosting têm os melhores valores de *Accuracy* (66%) e AUC (0.53), no entanto continuam a prever apenas a moda, pelo que mais uma vez o melhor modelo é o XGBoost.

Modelos	Accuracy	Precision	Recall	F1-Score	Support	AUC
Random Forest - 5 folds	0.62	-	-	-	-	0.51
Random Forest - 10 folds	0.62	-	-	-	-	0.50
Decision Tree	0.56	0.67 0.36	0.65 0.38	0.66 0.37	1660 842	0.51
Gradient Boosting	0.66	0.66 0.36	1.00 0.00	0.80 0.01	1660 842	0.53
AdaBoost	0.66	0.66 0.27	0.99 0.01	0.79 0.02	1660 842	0.53
XGBoost	0.62	0.67 0.36	0.84 0.18	0.75 0.24	1660 842	0.51

Tabela 6-Resultados Modelo 5

Modelo 6: Decidimos criar um modelo com as variáveis dummy referentes à variável *Ground*, pois pensamos que o tipo de chão pode afetar a velocidade a que os jogadores se

conseguem deslocar, tal como o serviço e o decorrer do jogo. Usamos as variáveis *DifNumberWins, DifRank, DifHeight, Ground_Clay* e *Ground_Hard*. Pela Figura 7, percebemos que as dummies contribuem, com cerca de 0.05 no total.

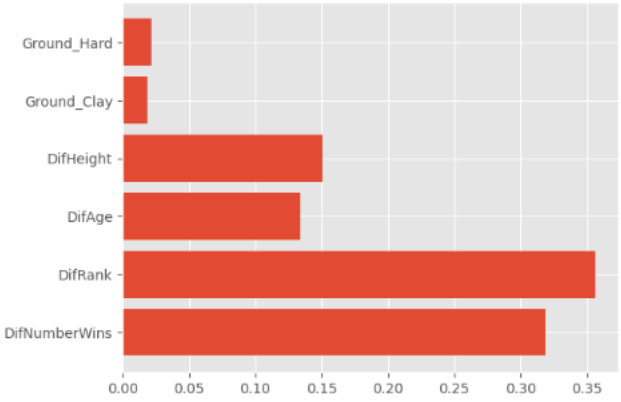


Figura 7-Feature Importance Modelo 6

Na Tabela 7, temos os resultados dos algoritmos. Estes são bastante interessantes, vemos que o valor de AUC do Gradient Boosting, AdaBoost e XGBoost são iguais (0.52), apesar de o valor de *Accuracy* dos dois primeiros ser maior que a do último (66% > 62%). No entanto, apenas o XGBoost não prevê a moda, logo este tem de ser considerado o melhor modelo.

Modelos	Accuracy	Precision	Recall	F1-Score	Support	AUC
Random Forest - 5 folds	0.61	-	-	-	-	0.49
Random Forest - 10 folds	0.60	-	-	-	-	0.49
Decision Tree	0.55	0.67 0.36	0.63 0.40	0.65 0.38	1654 848	0.52
Gradient Boosting	0.66	0.66 0.33	1.00 0.00	0.80 0.00	1654 848	0.52
AdaBoost	0.66	0.66 0.50	1.00 0.01	0.80 0.01	1654 848	0.52
XGBoost	0.62	0.66 0.36	0.85 0.16	0.75 0.22	1654 848	0.52

Tabela 7-Resultados Modelo 6

Modelo 7: Decidimos adicionar a variável *Prize* às variáveis anteriores, ficando com *DifNumberWins, DifRank, DifHeight, Prize, Ground_Clay* e *Ground_Hard* (Figura 8). Mais uma vez, *DifRank* é a mais relevante e as dummies são as menos relevantes.

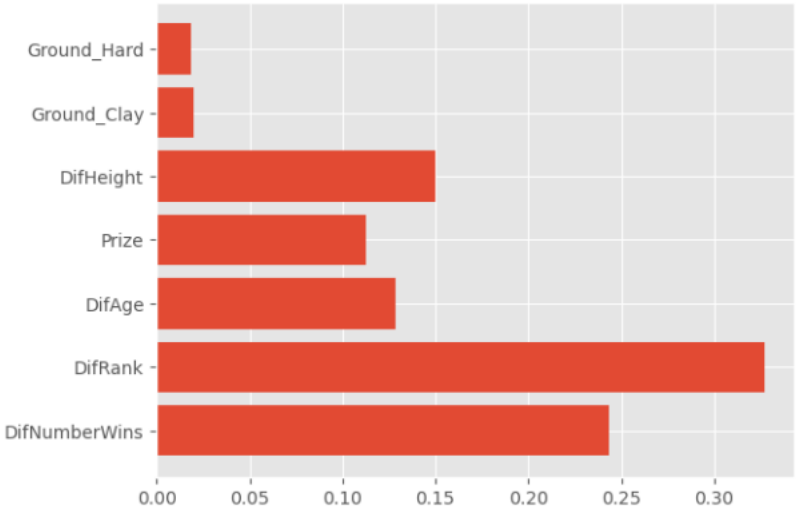


Figura 8-Feature Importance Modelo 7

Na Tabela 8, vemos que os valores de *Accuracy* e AUC, obtidos no Gradient Boosting e AdaBoost, são de 65% e 0.53, respetivamente, sendo os mais altos para o Modelo 7. No entanto, continuam a prever a moda, pelo que o XGBoost, ainda que tenha ligeiramente menor valor de *Accuracy* e AUC tem de ser considerado o melhor.

Modelos	Accuracy	Precision	Recall	F1-Score	Support	AUC
Random Forest - 5 folds	0.62	-	-	-	-	0.50
Random Forest - 10 folds	0.62	-	-	-	-	0.50
Decision Tree	0.55	0.66 0.36	0.66 0.36	0.66 0.36	1631 871	0.51
Gradient Boosting	0.65	0.65 0.26	0.99 0.01	0.79 0.01	1631 871	0.53
AdaBoost	0.65	0.65 0.75	1.00 0.00	0.79 0.01	1631 871	0.53
XGBoost	0.61	0.66 0.37	0.84 0.18	0.74 0.24	1631 871	0.51

Tabela 8- Resultados Modelo 7

Modelo 8: Incorporamos as variáveis *DifNumberWins*, *DifRank*, *DifAge*, *DifHeight*, *L_OR_R_Left-Handed*, *L_OR_R_Right-Handed*, *L_OR_R__Opponent_Left-Handed e L_OR_R__Opponent_Right-Handed*, *Ground_Clay* e *Ground_Hard*, com o objetivo de verificar se com maior número de variáveis se conseguiria melhorar de forma significativa os resultados. Na Figura 9, vemos que DifRank é a que tem mais relevância

e das dummies as com menos relevância são *L_OR_R_Left-Handed* e *L_OR_R_Right-Handed.*

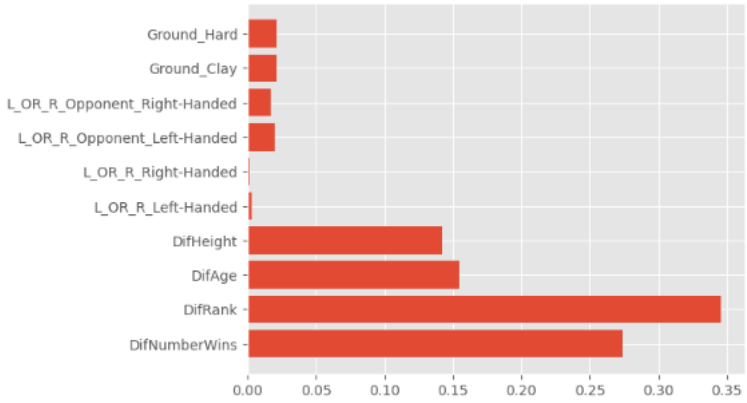


Figura 9-Feature Importance Modelo 8

Na Tabela 9, são apresentados os resultados, os melhores valores de *Accuracy* e AUC são obtidos no Gradient Boosting. Mas o valor de AUC do Adaboost e XGBoost são iguais. Mais uma vez, o XGBoost não prevê a moda, sendo, por isso, o melhor.

Modelos	Accuracy	Precision	Recall	F1-Score	Support	AUC
Random Forest - 5 folds	0.61	-	-	-	-	0.50
Random Forest - 10 folds	0.62	-	-	-	-	0.51
Decision Tree	0.55	0.67 0.35	0.65 0.37	0.66 0.36	1656 846	0.51
Gradient Boosting	0.66	0.66 0.29	1.00 0.00	0.80 0.00	1656 846	0.53
AdaBoost	0.66	0.65 0.75	1.00 0.00	0.80 0.01	1656 846	0.52
XGBoost	0.60	0.66 0.33	0.82 0.17	0.73 0.23	1656 846	0.52

Tabela 9-Resultados Modelo 8

Modelo 9: Igual ao Modelo 8, só que adicionamos a variável *Prize*, ficando com as variáveis *DifNumberWins, DifRank, DifAge, Prize, DifHeight, L_OR_R_Left-Handed, L_OR_R_Right-Handed, L_OR_R_Opponent_Left-Handed* e *L_OR_R_Opponent_Right-Handed, Ground_Clay* e *Ground_Hard* (Figura 10). A importância relativa das variáveis é mantida face ao modelo anterior.

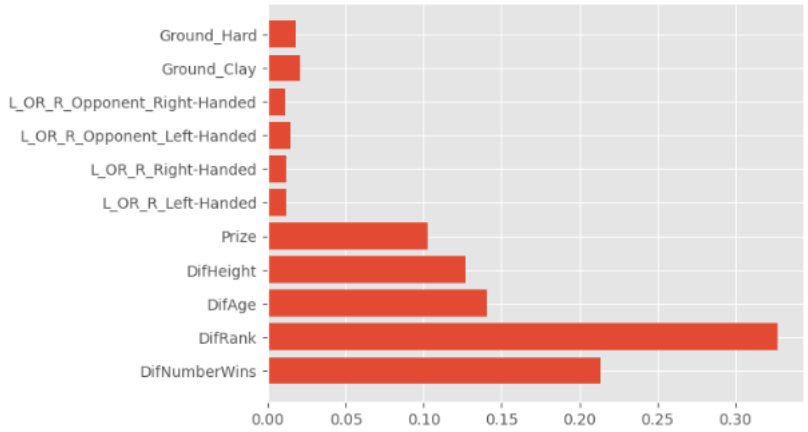


Figura 10-Feature Importance Modelo 9

Na Tabela 10, observamos que temos *Accuracy* de 65% no Gradient Boosting e AdaBoost, mas continuam a prever a moda. Pelo que, o XGBoost continua a ser o melhor algoritmo.

Modelos	Accuracy	Precision	Recall	F1-Score	Support	AUC
Random Forest - 5 folds	0.62	-	-	-	-	0.50
Random Forest - 10 folds	0.62	-	-	-	-	0.50
Decision Tree	0.56	0.66 0.37	0.66 0.37	0.66 0.37	1624 878	0.51
Gradient Boosting	0.65	0.65 0.56	1.00 0.01	0.79 0.02	1624 878	0.54
AdaBoost	0.65	0.65 0.55	0.99 0.01	0.79 0.02	1624 878	0.53
XGBoost	0.60	0.65 0.36	0.84 0.17	0.73 0.23	1624 878	0.52

Tabela 10-Resultados Modelo 9

Modelo 10: criamos a dummy *DifHands*, que vê se a mão utilizada pelo jogador principal é diferente do seu oponente, ou seja, compara *L_OR_R* e *L_OR_R_Opponent*. Tal como, está comprovado em diversos estudos existe um desequilíbrio quando jogadores com mãos dominantes diferentes se defrontam. Assim, utilizamos as variáveis *DifNumberWins, DifRank, DifAge, DifHeight, Prize e DifHands* (Figura 11). Destas variáveis *DifHands* é a que menos contribui.

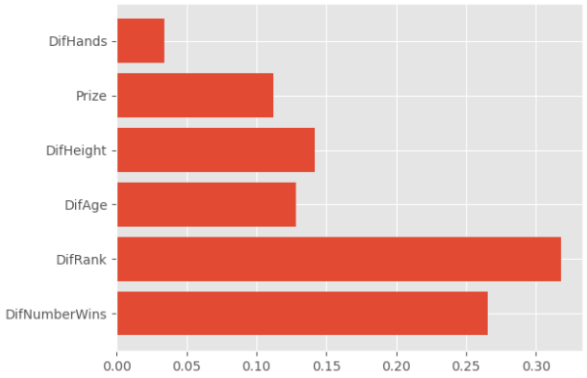


Figura 11-Feature Importance Modelo 10

Na Tabela 11, verificamos que alcançamos valores de *Accuracy* de 67% no Gradient Boosting e no AdaBoost, mas o último supera o primeiro ligeiramente no valor do AUC. No entanto, continuam a prever a moda, logo o XGBoost é o melhor.

Modelos	Accuracy	Precision	Recall	F1-Score	Support	AUC
Random Forest - 5 folds	0.61	-	-	-	-	0.51
Random Forest - 10 folds	0.61	-	-	-	-	0.51
Decision Tree	0.53	0.65 0.30	0.62 0.33	0.64 0.31	1680 822	0.48
Gradient Boosting	0.67	0.67 0.00	1.00 0.00	0.80 0.00	1680 822	0.53
AdaBoost	0.67	0.67 0.37	0.98 0.02	0.80 0.03	1680 822	0.54
XGBoost	0.62	0.67 0.34	0.84 0.17	0.75 0.23	1680 822	0.51

Tabela 11-Resultados Modelo 10

Modelo 11: Experimentamos com as variáveis *DifNumberWins*, *DifRank*, *DifAge*, *DifHeight*, *Prize*, *Ground_Clay*, *Ground_Hard* e *DifHands* (Figura 12). As dummy são as que menos contribuem para o modelo.

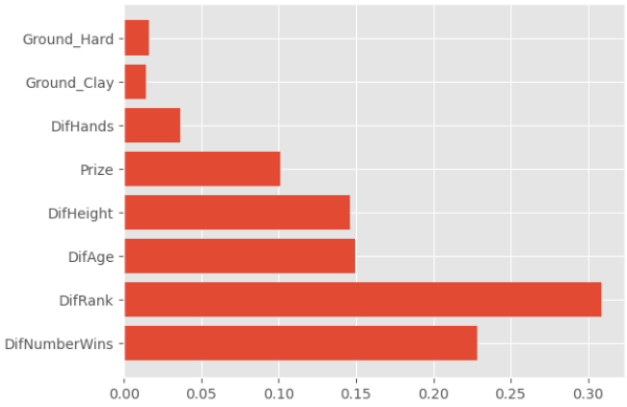


Figura 12-Feature Importance Modelo 11

Na Tabela 12, vemos que o Gradient Boosting tem melhor valor de *Accuracy*, mas o AUC é maior no AdaBoost. O XGBoost, apesar de ficar ligeiramente abaixo, não prevê a moda.

Modelos	Accuracy	Precision	Recall	F1-Score	Support	AUC
Random Forest - 5 folds	0.61	-	-	-	-	0.51
Random Forest - 10 folds	0.61	-	-	-	-	0.50
Decision Tree	0.54	0.66 0.32	0.63 0.36	0.64 0.34	1670 832	0.49
Gradient Boosting	0.67	0.67 0.00	1.00 0.00	0.80 0.00	1670 832	0.54
AdaBoost	0.66	0.67 0.40	0.99 0.02	0.80 0.04	1670 832	0.55
XGBoost	0.62	0.67 0.35	0.85 0.17	0.75 0.22	1670 832	0.52

Tabela 12-Resultados Modelo 11

Modelo 12: Considerando que em *HeightOpponent* falta uma parte considerável dos dados e imputá-los iriar artificializar em demasia o modelo, podendo alcançar resultados que nunca se iriam verificar na vida real, decidimos realizar um modelo sem *DifHeight*, ou seja, sem as variáveis relacionadas com a altura. Assim, ficamos com *DifNumberWins*, *DifRank*, *DifAge* e *Prize* (Figura 13). *DifRank* volta a ser a variável que contribui mais.

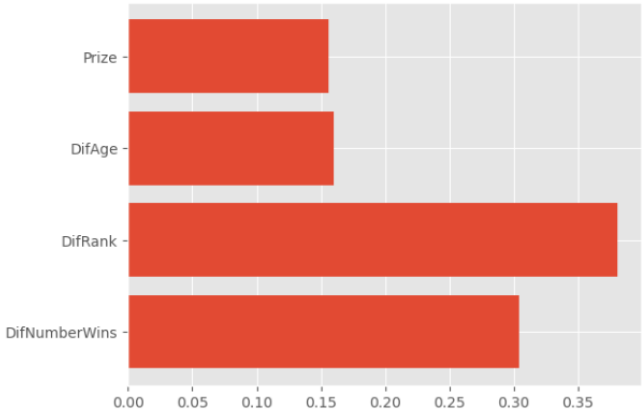


Figura 13-Feature Importance Modelo 12

Na Tabela 13, vemos que o Gradient Boosting e AdaBoost têm valores de *Accuracy* e AUC muito semelhantes, uma vez mais e que o XGBoost não prevê a moda.

Modelos	Accuracy	Precision	Recall	F1-Score	Support	AUC
Random Forest - 5 folds	0.60	-	-	-	-	0.50
Random Forest - 10 folds	0.60	-	-	-	-	0.51
Decision Tree	0.55	0.66 0.35	0.66 0.35	0.66 0.35	1641 861	0.50
Gradient Boosting	0.66	0.66 0.00	1.00 0.00	0.79 0.00	1641 861	0.53
AdaBoost	0.65	0.66 0.44	0.99 0.01	0.79 0.02	1641 861	0.54
XGBoost	0.61	0.66 0.35	0.86 0.14	0.74 0.20	1641 861	0.51

Tabela 13-Resultados Modelo 12

Modelo 13: Nesta fase, criamos a variável *DifNumGamesByYear*, pois um jogador que já tenha jogado mais jogos, poderá estar mais cansado, o que pode condicionar o número de sets e o seu desempenho. Decidimos, não inserir a variável *DifHeights* e utilizamos as variáveis *DifNumberWins*, *DifRank*, *DifAge*, *Prize* e *DifNumGamesByYear* (Figura 14). A importância relativa da variável criada é de cerca de 0.15.

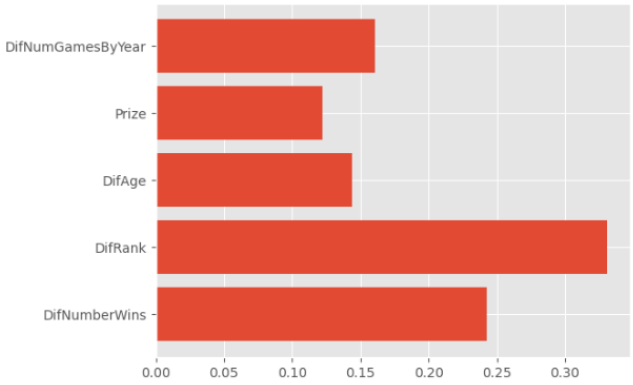


Figura 14-Feature Importance Modelo 13

Na Tabela 14, vemos que o Gradient Boosting e o Adaboost alcançam valores de *Accuracy* e AUC de 67% e 0.54, respetivamente. O XGBoost obtém nessa ordem 61% e 0.51, mas não prevê a moda.

Modelos	Accuracy	Precision	Recall	F1-Score	Support	AUC
Random Forest - 5 folds	0.63	-	-	-	-	0.53
Random Forest - 10 folds	0.63	-	-	-	-	0.53
Decision Tree	0.55	0.67 0.34	0.64 0.36	0.66 0.35	1669 833	0.50
Gradient Boosting	0.67	0.67 0.00	1.00 0.00	0.80 0.00	1669 833	0.54
AdaBoost	0.67	0.67 0.40	0.99 0.01	0.80 0.02	1669 833	0.54
XGBoost	0.61	0.67 0.34	0.83 0.18	0.74 0.23	1669 833	0.51

Tabela 14-Resultados Modelo 13

Modelo 14: é um modelo sem altura e com as variáveis *DifNumberWins*, *DifRank*, *DifAge* e *DifNumGamesByYear* (Figura 15). *DifRank* é a que contribui mais.

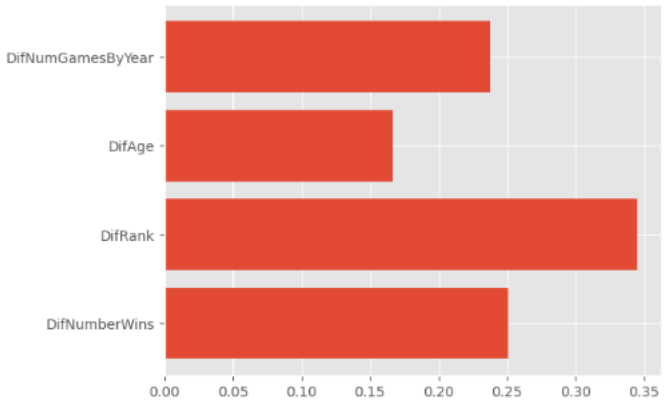


Figura 15-Feature Importance Modelo 14

Na Tabela 15, temos *Accuracy* de 65% para o Gradient Boosting e AdaBoost e de 60% para o XGboost, mas a diferença do valor de AUC entre todos não é muito elevada e o último não prevê a moda.

Modelos	Accuracy	Precision	Recall	F1-Score	Support	AUC
Random Forest - 5 folds	0.62	-	-	-	-	0.51
Random Forest - 10 folds	0.63	-	-	-	-	0.51
Decision Tree	0.56	0.66 0.37	0.65 0.38	0.66 0.38	1631 871	0.52
Gradient Boosting	0.65	0.65 0.00	1.00 0.00	0.79 0.00	1631 871	0.53
AdaBoost	0.65	0.65 0.20	1.00 0.00	0.79 0.00	1631 871	0.52
XGBoost	0.60	0.65 0.34	0.84 0.16	0.73 0.22	1631 871	0.51

Tabela 15-Resultados Modelo 14

Modelo 15: temos as variáveis *DifNumberWins*, *DifRank*, *DifAge*, *Prize*, *DifHands*, *Groun_Clay*, *Ground_Hard* e *DifNumGamesByYear* (Figura 16). As variáveis mais relevantes são *DifRank*, *DifNumberWins* e *DifNumGamesByYear*.

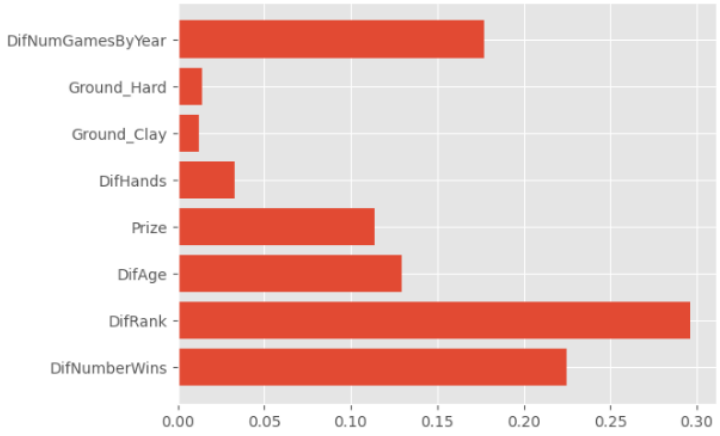


Figura 16-Feature Importance Modelo 15

Na Tabela 16, vemos que o valor de AUC mais elevado é de 0.53, no Gradient Boosting e AdaBoost. No entanto, o XGBoost, apesar de ter menor valor não prevê a moda.

Modelos	Accuracy	Precision	Recall	F1-Score	Support	AUC
Random Forest - 5 folds	0.63	-	-	-	-	0.49
Random Forest - 10 folds	0.62	-	-	-	-	0.49
Decision Tree	0.56	0.67 0.36	0.66 0.37	0.66 0.37	1647 855	0.52
Gradient Boosting	0.66	0.66 0.00	1.00 0.00	0.79 0.00	1647 855	0.53
AdaBoost	0.65	0.66 0.36	0.98 0.02	0.79 0.04	1647 855	0.53
XGBoost	0.61	0.66 0.35	0.83 0.17	0.74 0.23	1647 855	0.51

Tabela 16-Resultados Modelo 15

5. Avaliação

Relativamente aos resultados obtidos podemos começar por retirar algumas conclusões gerais acerca dos diferentes algoritmos:

- **Random Forest:** a nível de *accuracy* é bastante bom, considerando os valores obtidos em todos os modelos, estando sempre na casa dos 60%, mas o valor de AUC tende a ser bastante baixo, por vezes, abaixo de 0.50;
- **Decision Tree:** não prevê a moda, mas os seus valores de *Accuracy* e AUC são bastante baixos, tendo sempre os valores de *Accuracy* mais baixos que os restantes;

- **Gradient Boosting:** prevê a moda, mas é um dos melhores modelos a nível de *accuracy* e AUC;
- **AdaBoost:** muito parecido ao Gradient Boosting, tendo por vezes os mesmos resultados a nível das métricas;
- **XGBoost:** não prevê a moda e o valor de *Accuracy* e AUC não são muito inferiores aos do Gradient Boosting e AdaBoost.

Assim, em termos de performance se apenas considerarmos os valores de *Accuracy* e AUC, os melhores modelos tendem a ser o Gradient Boosting e AUC, no entanto estes ou não preveem ou preveem muitos poucos casos de 3 sets. Mas estes nunca poderão ser os algoritmos apresentados ao cliente, porque não garantem uma boa generalização em novos dados, pois apenas consegue prever a moda, coisa que mesmo sem algoritmo já seria relativamente fácil de prever. Deste modo, consideramos que o XGBoost é o melhor algoritmo, pois não prevê a moda e consegue ter valores de *Accuracy* e AUC bastante similares, ainda que inferiores.

Agora vamos realizar uma análise comparativa aos diferentes modelos para escolhermos qual o melhor modelo a apresentar ao nosso cliente considerando o objetivo de negócio proposto por ele, sendo que nos vamos focar no XGBoost, pois acreditamos na capacidade de generalização em novos dados.

- Comparando o Modelo **2** e **3**: estes são relativamente parecidos, mas relativamente ao XGBoost, apesar da *Accuracy* se manter em 61%, o valor de AUC aumenta de 0.50 (Modelo 2) para 0.52 (Modelo 3). Assim, avançamos para a continuação da comparação com o Modelo 3. E como a única diferença entre estes modelos é a variável Prize, concluímos que esta é relevante.

- Comparando os Modelos **3** e **4**: o Modelo 4, piorou ligeiramente, pelo que a adição das dummies de *L_OR_R* e *L_OR_R_Opponent* não são benéficas. Assim, continuamos com o Modelo 3 e concluímos que as variáveis dummy criadas a partir de *L_OR_R* e *L_OR_R_Opponent* não são muito relevantes, piorando ligeiramente os resultados inclusive.
- Comparando o Modelo **3** e **5**: vemos que os valores são muito parecidos para o XGBoost, apenas mudando em 0.01 o valor de *Accuracy* e AUC. Neste caso, decidimos atribuir maior relevância aos valores de AUC, pois a *Accuracy* é uma medida potencialmente enganosa quando existe um grande desequilíbrio entre classes. Assim, continuamos a escolher o Modelo 3. Mais uma vez, verifica-se que as dummies criadas a partir de *L_OR_R* e *L_OR_R_Opponent* não são muito relevantes e que inclusive o *Prize* ajuda a melhorar ligeiramente os resultados.
- Comparando o Modelo **3** e **6**: o Modelo 6 é melhor, pois apresenta valor de *Accuracy* de 0.62, ao ponto que o Modelo 3, apenas apresentava valor de 0.61 e o valor de AUC é de 0.52 nos dois modelos. Assim, avançamos com o Modelo 6. Podemos afirmar que as dummy criadas a partir de *Ground* são relevantes ajudando a melhorar o modelo.
- Comparando o Modelo **6** e **7**: o Modelo 7 apresenta valor de AUC e de *Accuracy* 0.01 mais pequenos que o Modelo 6, pelo que não pode ser considerado o melhor modelo.
- Comparando o Modelo **6** e **8**: o Modelo 8, apresenta o mesmo valor de AUC, mas o valor de *Accuracy* é inferior, pelo que não vais ser considerado. Mais uma vez, as dummies criadas a partir de *L_OR_R* e *L_OR_Opponent* não são benéficas.
- Comparando o Modelo **6** e **9**: o Modelo 9, apresenta os mesmos valores que o Modelo 8, continuando a ser pior que o Modelo 6.

- Comparando o Modelo **6** e **10**: o Modelo 10 apresenta o mesmo de valor de *Accuracy* que o Modelo 6, mas apresenta um valor de AUC ligeiramente menor, pelo que não vai ser considerado. A variável *DifHands* não é suficiente para melhorar significativamente os resultados neste caso.
- Comparando o Modelo **6** e **11**: os modelos apresentam os mesmos resultados de AUC e *Accuracy*, pelo que deixamos a sua comparação para depois. A adição de *DifHands* e *Prize* é benéfica, neste caso.
- Comparando os Modelos **6, 11** e **12**: o Modelo 12, não vai ser considerado, pois apresenta valores de AUC e *Accuracy* ligeiramente inferiores, talvez devido à simplificação do modelo.
- Comparando os Modelos **6, 11** e **13**: o Modelo 13, encontra-se na mesma situação que o Modelo 12, pelo que não vai ser considerado.
- Comparando os Modelos **6, 11 e 14**: o valor de AUC é 0.02 inferior ao dos Modelos 6 e 11 e 0.01 inferior no valor de AUC.
- Comparando os Modelos **6, 11** e **15**: o modelo 15, está na mesma situação que os Modelos 12 e 13.

Agora vamos perceber entre o Modelo **6** e **11**, qual é o melhor. Para isso vamos estudar as variáveis que os compõem. O Modelo 6 é composto por 6 variáveis: *Ground_Hard*, *Ground_Clay*, *DifHeight*, *DifAge*, *DifRank* e *DifNumberWins*. O Modelo 11, por sua vez, é compost por 8 variávies: *Ground_Hard, Ground_Clay, DifHands, Prize, DifHeight, DifAge, DifRank* e *DifNumberWins.*

Deste modo, consideramos que o modelo que vai mais de acordo com os interesses do nosso cliente seria o Modelo 11, não só por ser o com melhor performance ao nível do XGBoost, mas por ter em conta mais variáveis que consideramos bastante relevantes. Por exemplo, jogadores cuja mão dominante é diferente pode afetar a condução do jogo e

consequentemente o resultado deste, como vários estudos comprovam e o *Prize* pode atrair jogadores mais conceituados se o seu valor for elevado, o que pode tornar a competição mais desafiante, tal como pode levar a uma tentativa de melhor desempenho por parte dos jogadores mais novos para ficarem conhecidos pelas habilidades no desporto, atraindo fama e dinheiro, beneficiando tanto os jogadores como treinadores.

6. Implementação

A proposta final que temos para a Confederação Brasileira de Ténis vai permitir que o nível dos treinadores e jogadores brasileiros aumente, atraindo visibilidade e investidores para o seu país e talento nacional.

Os jogadores devem treinar em diferentes pisos, pois cada tipo de piso oferece condições de jogo distintas. Por exemplo, o comportamento da bola altera-se, podendo ser mais rápida ou não, logo o tempo de reação dos jogadores é essencial. Diferentes materiais condicionam a velocidade dos jogadores, devido a uma força de resistência denominada atrito, que varia consoante o material em causa. Além disso, o estilo de jogo/táticas adotadas podem beneficiar ou não consoante o tipo de chão, ou seja, depende se o comportamento do jogador é mais ofensivo ou defensivo. Deste modo, quanto mais experiência tiverem em diferentes tipos de pisos, mais facilmente adaptam o seu estilo de jogo, ainda que seja necessário ter atenção ao facto de certos tipos de piso serem mais escorregadios, logo mais propensos a lesões.

A diferença de mão dominantes, implica uma diferença na forma como pega na raquete e a utiliza nas diferentes jogadas realizadas, podendo variar a forma como serve e remata, e consequentemente a sua estratégia de jogo. A consciência das diferenças do seu oponente pode levar a uma preparação específica para determinado jogo e a utilização de táticas diferentes.

Torneios com prémios mais elevados, podem levar os novos talentos a estarem mais motivados, porque vão conseguir competir contra os seus eventuais jogadores no topo do rank, porque normalmente quanto maior o prémio, mais celebrado é o torneio, logo existe uma maior movimentação de pessoas afluentes.

Nos treinos, também devem competir contra jogadores de alturas diferentes, pois uma diferença de alturas implica diferentes alcances e coberturas do recinto e é importante adotarem estratégias que lhes permitam não se sentirem dominados pelo adversário. Além disso, pode ser uma experiência interessante treinarem com jogadores de idades, *ranks* e experiências diferentes, pois podem aprender com a experiência dos outros, o seu estilo de jogo, possibilidades de evolução a nível de táticas, como lidar com a pressão e se podem adaptarem a diferentes adversários.

Deste modo, é essencial que os treinadores criem treinos personalizados, após perceberem as falhas ou aspetos a melhorar dos seus jogadores, tendo em conta os objetivos e necessidades do jogador e o seu estilo de jogo.

Os treinadores devem implementar planos de treino com um cronograma, para que os jogadores consigam alcançar os seus objetivos sem desmotivarem e monitorizá-los durante todo o processo para irem adaptando os treinos. Devem também monitorizar o desempenho nos jogos oficiais para perceberem como funciona o plano em ambientes com maior pressão.

7. Conclusão

A abordagem estruturada da metodologia CRISP-DM foi essencial para a condução da análise dos dados, pois a compreensão das diversas vertentes problema e das variáveis em causa foram fundamentais para que conseguíssemos chegar a um bom modelo e a uma proposta que satisfizesse os desejos do nosso cliente.

Ao longo desta análise foi interessante perceber como variáveis à partida sem grande relevância, condicionam o desempenho dos jogadores, como por exemplo, o tipo de chão ou a mão dominante utilizada ser diferente entre jogadores.

Deste modo, podemos afirmar que os resultados obtidos num jogo de ténis dependem de características internas e externas ao jogador, ou seja, dependem de características físicas, táticas, experiência, adaptabilidade e tipo de competição.

Assim, os treinadores são essenciais para que os jogadores consigam ter melhor desempenho, devendo criar-lhes um treino que vá de encontro aos seus objetivos e que os desafie constantemente, estando aptos para competirem com jogadores mais experientes e de renome.

No entanto, estas descobertas estão condicionadas pelo *dataset* fornecido e por todas as decisões tomadas pela equipa de Ciência de Dados ao longo das etapas, logo a não imputação de diversos valores pode ter afetado os resultados, ainda que tenhamos tentado que eles ficassem o mais reais possíveis.

Não obstante, as descobertas e as recomendações dadas à Confederação Brasileira de Ténis podem ajudar todos os jogadores e treinadores que lhes estão associados, fazendo com que o Brasil ganhe visibilidade e consequentemente investidores.

Para além disso, poder-se-ia também experimentar variáveis novas, como por exemplo a força média dos serviços do jogador, estudar outros modelos possíveis como SVM e algoritmos de *Deep Learning* como Multilayer Perceptron (MLP). Como trabalho futuro, também poderia ser explorada a metodologia SCRUM, e assim, complementar a metodologia CRISP-DM de forma a permitir adaptabilidade, colaboração multidisciplinar e a entrega contínua de resultados aos clientes, sempre tendo transparência em todo o processo otimizando o projeto de análise de dados.

Assim, em ações futuras poderíamos explorar os dados mais a fundo, complementando-os ao tentar encontrar padrões ou variáveis novas, igualmente ou mais relevantes do que as estudadas.

8. Referências Bibliográficas

ATP. (n.d.). *History*. Retrieved from atptour: https://www.atptour.com/en/corporate/history

ATP. (n.d.). *Senior Leadership Team*. Retrieved from ATP: https://www.atptour.com/en/corporate/management

B., E. (2006, 05). Retrieved from National Library of Medicine: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2577481/

Breznik, K. (2013, 06). Retrieved from National Library of Medicine: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3761843/

Hadlich, G. (n.d.). *How Does Tennis Scoring Work? (By Former PRO)*. Retrieved 04 13, 2023, from mytennishq: https://mytennishq.com/tennis-scoring-rules-the-ultimate-guide-explained/

IrvingWeekly Staff. (2022, 06 22). *Height A Great Factor In Tennis?* Retrieved from IrvingWeekly: https://www.irvingweekly.com/s/4280/Is-Height-a-Great-Factor-in-Tennis.php

Kothari, R. (2022, 02 24). *4 Types of Surfaces Used for Tennis Court*. Retrieved 04 14, 2023, from Rubcorp: https://www.rubcorp.com/types-of-tennis-court-surfaces/

Miller, S. (2018, 07 04). *Repleto de 'gigantes', tênis também tem espaço para 'baixinhos'*. Retrieved from THE NEW YORK TIMES: https://www1.folha.uol.com.br/esporte/2018/09/repleto-de-gigantes-tenis-tambem-tem-espaco-para-baixinhos.shtml

Rio Open. (n.d.). *O Torneio*. Retrieved from Rio Open: https://rioopen.com/o-torneio/

Sackmann, J. (2017, 07 04). *how much does height matter in mens tennis*. Retrieved from tennisabstract: http://www.tennisabstract.com/blog/2017/09/04/how-much-does-height-matter-in-mens-tennis/

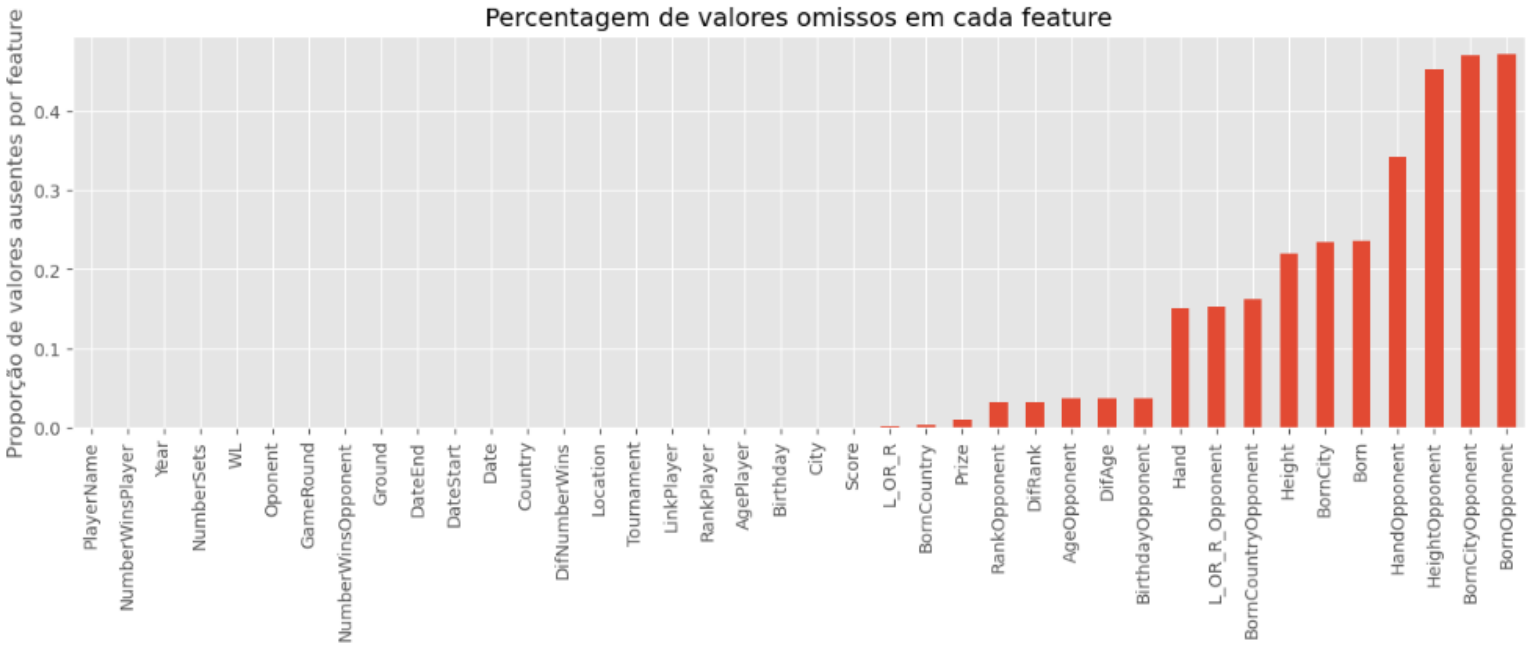
SteveGTennis. (2012, 07 17-23). *2012 Challenger Qualifying Campinas*. Retrieved from https://www.stevegtennis.com/h2h-predictions/2012-challenger-qualifying-campinas/

Surfaces Used for Tennis Court. (2022, 02 24). Retrieved from RubCorp: https://www.rubcorp.com/types-of-tennis-court-surfaces/

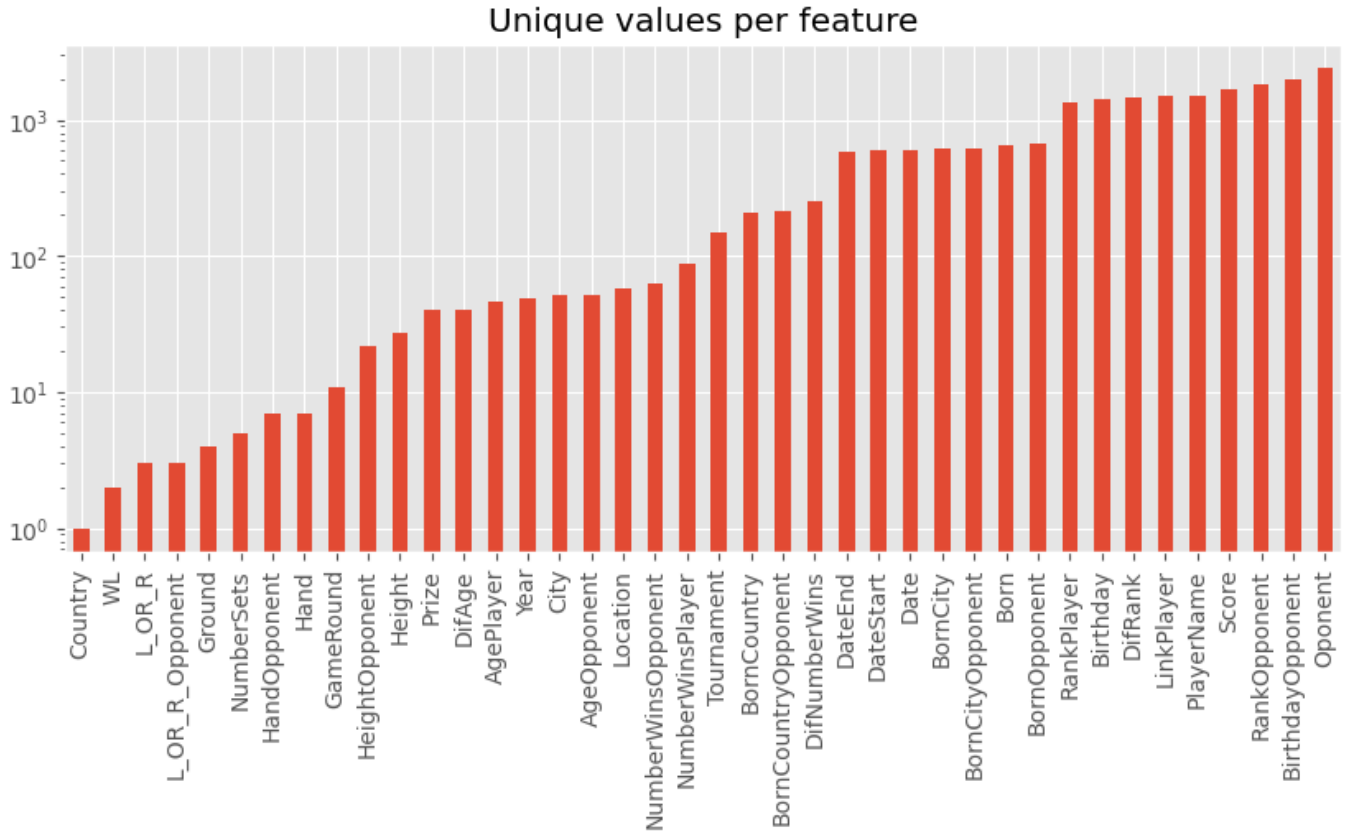
Wikipédia. (2023, 06 1). *ATP Rankings*. Retrieved from https://en.wikipedia.org/wiki/ATP_rankings

9. Anexos

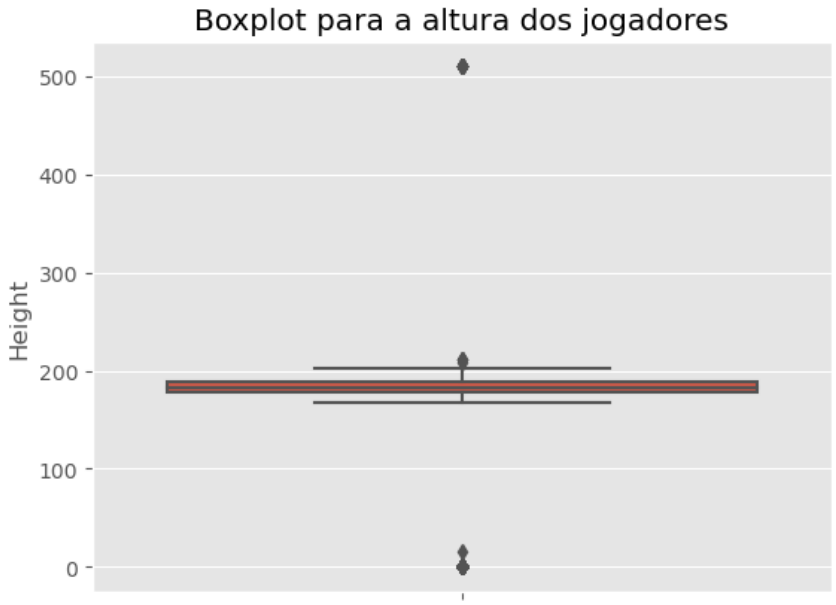
Anexo 1 – Valores omissos (em percentagem) para ATP Brasil



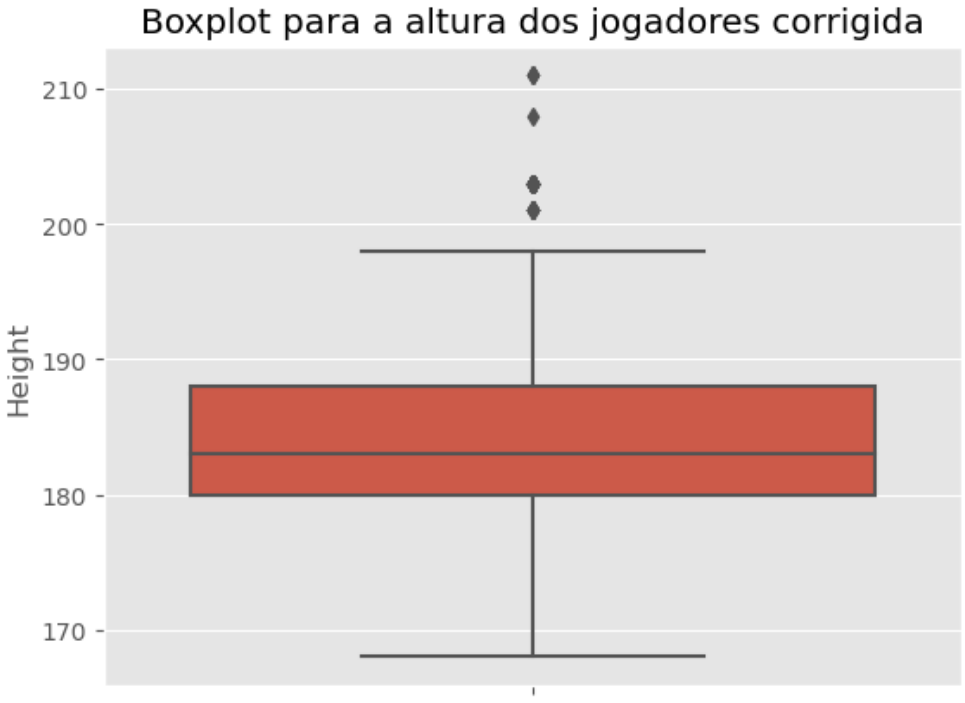
Anexo 2 – Gráfico com os valores únicos por variável



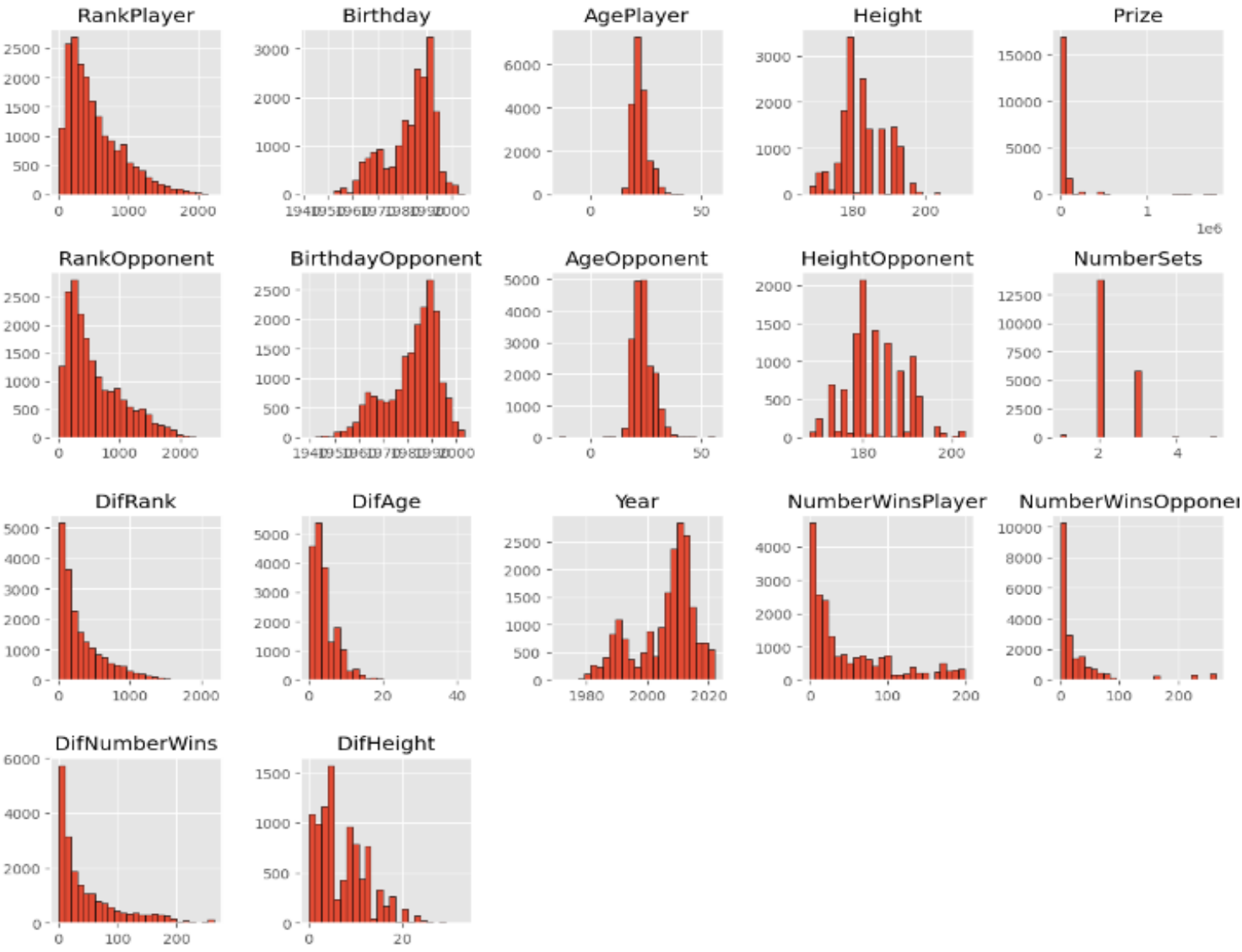
Anexo3 – Boxplot para a altura dos jogadores



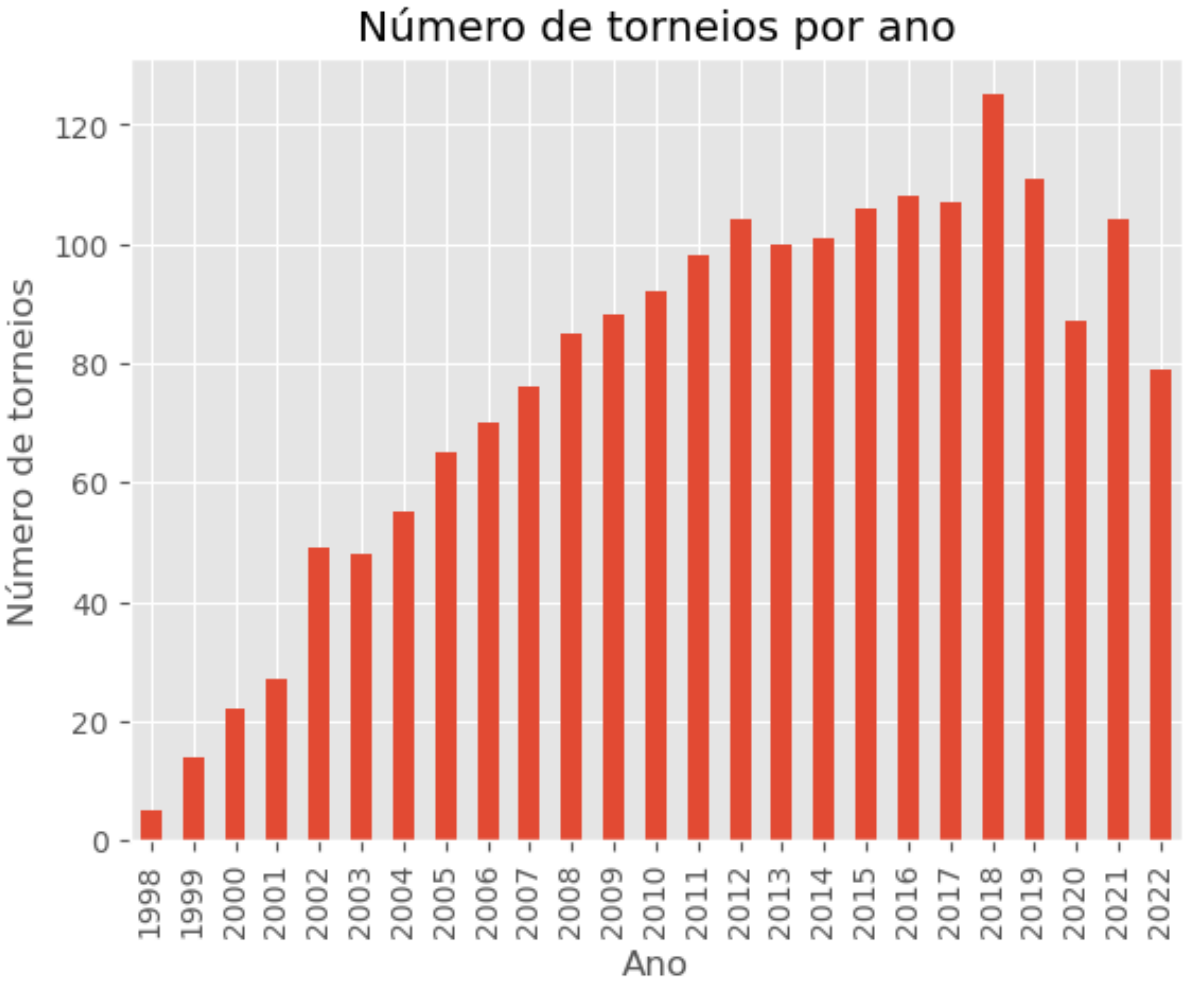
Anexo 4 - Boxplot para a altura dos jogadores corrigida



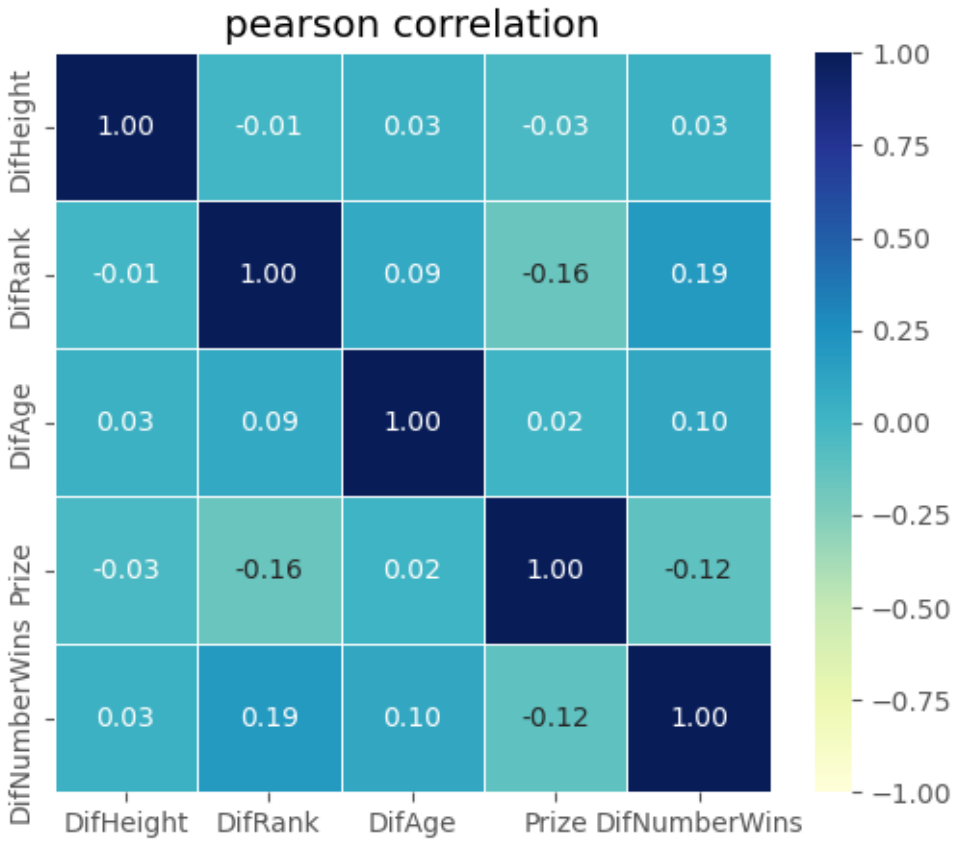
Anexo 5 - Conjunto de histogramas sobre as variáveis numéricas



Anexo 6 – Torneios por ano



Anexo 7 – Correlações de Pearson



Anexo 8 – Correlações de V de Cramer para as variáveis categóricas

	NumberSets	L_OR_R_Left-Handed	L_OR_R_Right-Handed	L_OR_R_Opponent_Left-Handed	L_OR_R_Opponent_Right-Handed	Ground_Clay	Ground_Hard
NumberSets	1.000000	0.026569	0.026012	0.039089	0.038210	0.008337	0.006735
L_OR_R_Left-Handed	0.026569	1.000000	0.998780	0.020957	0.021562	0.005182	0.005309
L_OR_R_Right-Handed	0.026012	0.998780	1.000000	0.021335	0.021940	0.004540	0.004699
L_OR_R_Opponent_Left-Handed	0.039089	0.020957	0.021335	1.000000	0.998070	0.049688	0.043919
L_OR_R_Opponent_Right-Handed	0.038210	0.021562	0.021940	0.998070	1.000000	0.050605	0.044794
Ground_Clay	0.008337	0.005182	0.004540	0.049688	0.050605	1.000000	0.949580
Ground_Hard	0.006735	0.005309	0.004699	0.043919	0.044794	0.949580	1.000000