INSTITUTO UNIVERSITÁRIO DE LISBOA

Projeto Aplicado em Ciência de Dados I

PREVISÃO DO NÚMERO DE SETS EM JOGOS DO ATP TOUR

Docentes:

Diana Mendes e Sérgio Moro

Grupo 9, CDB2

Francisco Rodrigues 105427

Margarida Carvalho 104765

Maria Margarida Pereira 105877

Simão Fonseca 105251

maio de 2023

Agradecimentos

Gostaríamos de aproveitar este momento para expressar os nossos sinceros agradecimentos ao professor Sérgio Moro e à professora Diana Mendes pelo acompanhamento, dedicação e contribuições inestimáveis ao nosso crescimento académico e pessoal.

Tanto o professor Sérgio Moro quanto a professora Diana Mendes, estiveram sempre disponíveis para nos ajudar, demonstrando um compromisso notável.

O amplo conhecimento em Ciência de Dados, bem como o entusiasmo que compartilharam connosco, proporcionou uma experiência completa e enriquecedora, permitindo explorar e compreender conceitos fundamentais na nossa área.

Este projeto não teria sido possível sem a orientação e suporte dos professores, as sugestões e feedbacks contribuíram para a qualidade e rigor dos nossos resultados.

Muito obrigado pela orientação, confiança e apoio ao longo deste trabalho.

ÍNDICE

Introdução

No âmbito da Unidade Curricular Projeto Aplicado em Ciência de Dados I realiza-se um trabalho de grupo que tem como objetivo explorar os dados e prever o n.º de sets para a conclusão de um jogo de ténis profissional (ranking ATP). Em termos de ferramentas, o projeto será implementado recorrendo ao Jupyter Notebook e à linguagem de programação Python.

O presente relatório visa enunciar o problema em estudo e respetivos dados utilizados, abordar os aspetos mais relevantes sobre as decisões tomadas, bem como experiências e testes realizados, tendo em consideração os ambientes de desenvolvimento e teste utilizados.

Neste sentido, o dataset que iremos trabalhar refere-se aos ATP Players, no qual estão todas as partidas individuais de 10 361 jogadores profissionais de ténis masculinos (top 500 jogadores que jogaram entre 28/03/1973 e 14/02/2022).

O desenvolvimento do nosso trabalho segue a metodologia CRISP-DM, que significa Cross-Industry Standard Process for Data Mining. Fornece uma visão geral do ciclo de vida dos dados, de maneira a obter um conhecimento mais aprofundado, através da limpeza, preparação, exploração e visualização dos mesmos.

O CRISP-DM pode ser executado de maneira não estrita (pode se mover para frente e para trás entre diferentes fases). Por definição, esta metodologia é formada por seis fases: Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation e Deployment, tal como apresentado na figura 1.

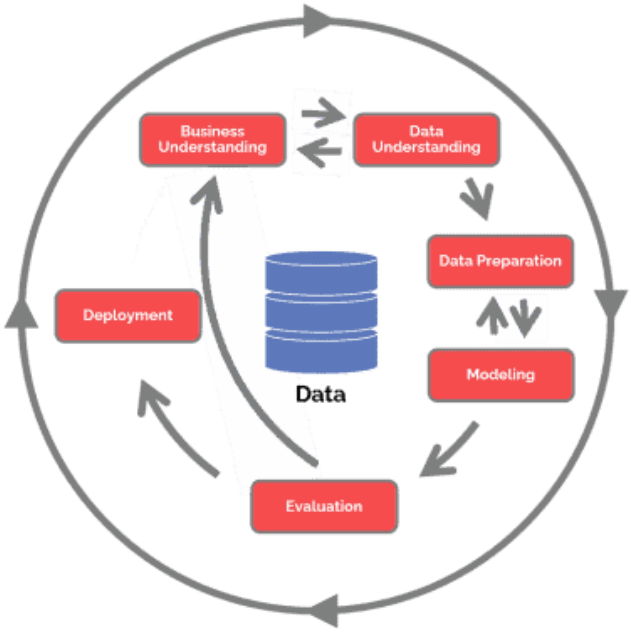

Figura 1 – Fases da metodologia CRISP-DM

1. Business Understanding

Primeiramente, de maneira a compreender a nossa base de dados ATP (Association of Tennis Professionals), começámos por perceber a sua origem, bem como a evolução do ténis profissional.

Em 1972, durante a primeira semana do US Open em Forest Hills, os principais profissionais uniram forças para criar a Associação de Profissionais de Ténis, para proteger os interesses dos tenistas profissionais. Assim, a ATP tornou-se o órgão regulador dos circuitos de ténis profissionais masculinos – o ATP Tour, o ATP Challenger Tour e o ATP Champions Tour. Para regular a competição, um dos atos iniciais da organização foi o estabelecimento de um sistema de classificação por computador que fornecia uma análise justa do desempenho de um jogador. O ranking ATP começou a 23 de agosto de 1973 e continua até hoje como o sistema de classificação oficial do ténis profissional masculino.

A classificação ATP é uma lista que classifica os jogadores profissionais masculinos com base nos seus resultados e prestações em torneios individuais e de duplas. Os jogadores recebem pontos com base no seu desempenho em cada torneio, e a classificação é atualizada semanalmente. A ATP Tour é um conjunto de torneios profissionais de ténis masculino, composto por uma série de torneios realizados ao longo do ano em diferentes locais do mundo, e o desempenho dos jogadores vai ser contabilizado para o seu ranking ATP.

Em relação a um jogo de ténis, este é constituído por sets. Um set de ténis é uma das unidades que compõem um jogo completo. É um conjunto de jogos que termina quando um jogador ganha pelo menos seis jogos e tem uma vantagem de pelo menos dois jogos em relação ao seu oponente. Por exemplo, se um jogador ganhar seis jogos e o outro jogador ganhar quatro jogos, o primeiro jogador ganha o set. Se o placar estiver empatado em 6 a 6, é jogado um tie-break, que é um tipo especial de jogo que determina o vencedor do set.

Cada jogo é iniciado com um dos jogadores fazendo um serviço, que corresponde ao "lançamento" da bola para o campo do oponente. O jogador que faz o serviço alterna com o seu oponente a cada dois jogos. Para ganhar um jogo, um jogador precisa vencer em pelo menos quatro pontos, com uma vantagem de pelo menos dois pontos em relação ao oponente. Os pontos são marcados em uma sequência que vai de zero a quinze a trinta a quarenta e finalmente ao jogo, que é quando um jogador ganha por vantagem igual ou superior a dois pontos.

Normalmente, um jogo completo de ténis é jogado à melhor de três ou melhor de cinco sets. Isso significa que o primeiro jogador a vencer dois ou três sets, dependendo das regras do torneio, é declarado o vencedor. Torneios de Grand Slam, como o Open da Austrália, o Open

de França, Wimbledon e o Open dos EUA, têm jogos masculinos que são jogados à melhor de cinco sets, enquanto a maioria dos outros torneios tem jogos masculinos jogados à melhor de três sets.

No nosso caso, uma vez que o objetivo é estudar e tentar prever os sets em torneios realizados na China, sabemos que o máximo de sets que pode existir por jogo é 3 sets, uma vez que na China não existe nenhum torneio de Grand Slam.

Sendo que queremos prever o número de sets, o problema é considerado um problema de classificação, uma vez que estamos a tentar atribuir uma classe ou categoria específica a cada instância. Neste caso, a classe ou categoria seria o número de sets (por exemplo, 2 sets, 3 sets, etc.). Ou seja, estamos a lidar com uma variável discreta e limitada de categorias possíveis.

Deste modo, queremos construir um modelo que seja capaz de classificar corretamente o número de sets num jogo de ténis.

1.1 Dicionário de termos

- Set – Divisão de uma partida de ténis. As partidas são jogadas à melhor de 3 ou 5 sets, ou seja, quem ganhar dois ou três sets ganha a partida.

- Jogo – Divisão dos sets. Para ganhar um set é necessário chegar aos 6 jogos, ou seja, fazer Game Point 6 vezes, com uma diferença de pelo menos 2 jogos, caso tal não aconteça vai-se a TieBreak.

- Serviço – Primeira jogada feita num ponto. O jogador que serve, lança a bola ao ar e atira-a de modo que esta aterre no quadrado de serviço do lado oposto e diagonal a si, do campo. Caso o jogador falhe este primeiro serviço, tem direito a uma segunda tentativa.

- Tiebreak – Ocorre quando existe um empate 6-6 no set, em que o jogador só ganha se fizer uma diferença de dois jogos, 8-6 por exemplo, ficando o set registado como 7-6.

- Court – Superfície onde os jogadores jogam, consiste num campo retangular, 23.77m por 8.23m, dividido por uma rede de 1.07m e delimitado por várias linhas de definem onde se pode servir e os limites de onde a bola pode bater. Este pode ter diferentes pisos: piso duro, relva e terra batida, só para mencionar alguns

- Pontuação – Tem um sistema único em que os pontos são contados da seguinte maneira: Love; 15; 30; 40 e Game Point. Ganha o jogo quem fizer Game Point primeiro.

1.2 Objetivo

Este trabalho tem como objetivo prever o número de sets num jogo de ténis.

O objetivo é desenvolver modelos que sejam capazes de analisar diversas variáveis relevantes, como o ranking dos jogadores, histórico de jogos anteriores, tipo de superfície e condições climáticas, e gerar uma previsão do número de sets que provavelmente será jogado numa partida de ténis.

Nesta fase do trabalho, definimos também qual seria o negócio no qual o nosso modelo estaria incluído. Existiam inicialmente duas opções: anúncios publicitários e apostas desportivas.

Num jogo de ténis, existem pausas. Sendo que a cada 3 jogos de serviços, existe uma pausa de 60 segundos. Esta pausa, mais os 20 segundos que os jogadores têm para iniciar um jogo de serviço, perfila um total de 80 segundos. É nesta altura que são passados os anúncios de televisão, durante estes 80 segundos.

O problema desta ideia, é que o nosso modelo prevê sets e tanto quanto sabemos, é possível ter um jogo de três sets e acabar em 6-0, 0-6, 6-0. Ora, isto corresponderia a sensivelmente seis blocos de espaços publicitários possíveis.

Agora, imaginemos um jogo que acabe com os parciais de 7-6, 6-7, 7-6. Significa então que este jogo também teve 3 sets, mas que ao contrário do exemplo anterior, este teria entre 9 e 10 espaços publicitários.

Caso o objetivo do trabalho, fosse a criação de um modelo que previsse o número de jogos num jogo de ténis, então sim, faria sentido incutir o modelo para anúncios publicitários.

Por esta grande falha é que decidimos não avançar com a ideia dos anúncios publicitários.

Deste modo, decidimos avançar com a ideia das apostas desportivas.

Nas atuais casas desportivas, em relação ao número de sets, existem 2 modalidades diferentes onde o nosso modelo irá ser extremamente útil:

- Número total de sets jogados num jogo: Neste tipo de modalidade, os apostadores apostam no número exato de sets num jogo;

- Acima/Abaixo: Este é um tipo de modalidade em que se aposta se um jogo tem acima ou abaixo de um determinado número de sets proposto por uma casa de apostas (por norma, em jogos onde o máximo de sets são 3, as casas de apostas usam 2.5 como valor de referência). Por exemplo, caso um apostador acha que determinado jogo irá ter 3 sets, então deverá apostar em "+2.5", mas caso acha que esse jogo terá menos que 3 jogos, deverá apostar em "-2.5".

Esta previsão pode auxiliar os apostadores a tomar decisões informadas e estratégicas, considerando a duração esperada do jogo. No entanto, é relevante salientar que a previsão do número de sets numa partida de ténis pode ser afetada por uma série de fatores imprevisíveis, como o desempenho individual dos jogadores e o contexto específico do jogo, o que requer uma análise cuidadosa e consideração dos riscos envolvidos nas apostas desportivas.

2. Data Understanding

Esta fase do projeto foi dedicada à segunda fase do processo CRISP-DM, o Data Understanding. Esta fase é caracterizada pela exploração e compreensão da base de dados, das variáveis existentes e no número de instâncias existentes, para ser possível começar a desenvolver uma estratégia para a fase seguinte.

Neste capítulo, foram realizadas análises e investigações sobre os dados disponíveis, com o objetivo de entender a sua estrutura, características e qualidade. Foram examinadas as variáveis relevantes, a sua distribuição e possíveis problemas de qualidade, como dados em falta ou inconsistentes. Além disso, foram utilizadas técnicas estatísticas e de visualização de dados para obter uma compreensão aprofundada dos padrões e tendências presentes na base de dados. Através desta etapa do projeto, foi então possível obter informações importantes para os passos seguintes do projeto, onde foram realizadas as limpezas, criação de variáveis, criação de modelos, entre outros.

2.1 Importação da base de dados

O projeto foi realizado em Python, e para uma melhor manipulação dos dados, era necessário que a base de dados estivesse num formato .csv. Uma vez que a base de dados fornecida vinha em formato .json, foi necessário, antes de iniciar a fase em Python, a utilização de um outro software para fazer essa conversão. Deste modo, foi utilizada uma ferramenta que já tinha sido utilizada em outras unidades curriculares, o MongoDB, que é um software utilizado para gestão de bases de dados NoSQL e muito útil para ambientes de Big Data. Depois de convertido o ficheiro para formato .csv, foi então utilizado o Python para o resto do projeto.

Foi criado um dataframe inicial com os dados fornecidos pelos docentes, que serviu para os outros passos seguintes.

	PlayerName	Born	Height	Hand	LinkPlayer	Tournament	Location	Date	Ground	Prize	GameRound	GameRank	Oponent	WL	Score
0	Novak Djokovic	Belgrade, Serbia	188.0	Right-Handed, Two-Handed Backhand	https://www.atptour.com/en/players/novak-djoko...	Davis Cup Finals	Madrid, Spain	2021.11.22 - 2021.12.05	Hard	NaN	Round Robin	118	Dennis Novak	W	63 62
1	Novak Djokovic	Belgrade, Serbia	188.0	Right-Handed, Two-Handed Backhand	https://www.atptour.com/en/players/novak-djoko...	Davis Cup Finals	Madrid, Spain	2021.11.22 - 2021.12.05	Hard	NaN	Semi-Finals	30	Marin Cilic	W	64 62

Figura 2 - Visualização das duas primeiras linhas do dataframe

2.2 Variáveis em estudo

PlayerName – Indica o primeiro e último nome do jogador principal da partida.

Born - Assinala a cidade e/ou país onde o jogador principal é natural.

Height - Designa a altura dos jogadores, em centímetros.

Hand - Indica qual a mão que o jogador utiliza para jogar, sendo que este pode ser destro - *Right-Handed*, canhoto - *Left-Handed* ou ambidestro – *Ambidextrous*; e que mão utiliza para executar serviço, backhand de uma mão - *One-Handed Backhand*, backhand de duas mãos - *Two-Handed Backhand* ou backhand desconhecida - *Unknown Backhand*.

LinkPlayer – Fornece o link do perfil do jogador principal no website ATP Tour.

Tournament – Expõe o nome do torneio. Esta variável contém vários nomes de torneios que se encaixam nestas 8 grandes categorias:

- **Grand Slams** - Os quatro maiores e mais prestigiosos torneios de ténis: Australia Open; Roland Garros; Wimbledon; US Open, e que oferecem o maior número de pontos, 2000, ao vencedor
- **ATP World Tour Masters 1000** – A seguir aos Grand Slams estes são os 9 mais prestigiosos que são realizados em várias localizações à volta do mundo sendo algumas delas Monte-Carlos, Miami e Shanghai. Tal como o nome indica, o vencedor recebe 1000 pontos.
- **ATP Finals** – Também conhecida como Nitto ATP Finals, esta pega nos 8 melhores jogadores da temporada para competirem pelos 1500 pontos que esta oferece.
- **ATP Tour 500** – Abaixo dos Masters temos os 13 torneios que compõem este circuito que, tal como o nome diz, oferecem 500 pontos ao vencedor
- **ATP Tour 250** – O circuito de nível mais baixo de todas as ATP Tours, existem cerca de 40 torneios deste tipo anualmente que dão ao campeão 250 pontos.
- **ATP Challenger Tour** – Torneios que oferecem aos jogadores dos rankings mais baixos uma oportunidade de subir de nível e qualificarem-se para as ATP Tours. Os pontos variam entre 125 e 80.
- **ITF Future Tour** - Organizado pela Federação Internacional de Ténis, este é o primeiro passo para os profissionais, pois é aqui que os jogadores conseguem os seus primeiros pontos para entrar no ranking e puderem competir noutras Tours.
- **Davis Cup** – Servem como um mundial de ténis onde os jogadores representam o seu país nesta competição, mas nenhuns pontos para o ranking são atribuídos aos jogadores.

Location – Indica a localização onde o torneio se realiza. Esta irá possuir um único valor, China.

Date – Revela o período de realização do torneio. Esta vem no formato ano.mês.dia - ano.mês.dia (ex: 2015.10.05 – 2015.10.11), em que a primeira data representa o início do torneio e a segunda data o seu fim.

Ground – Demonstra o tipo de piso do terreno onde é jogado o torneio. Esta variável apresenta quatro valores distintos: Hard (piso duro), Clay (terra batida), Carpet (relva sintética) e Grass (relva natural).

Prize – Denota o prémio monetário que o jogador principal recebe, em dólares. Como estes valores são para o ano em que se realizou o torneio, não têm em conta a inflação ao longo dos anos

GameRound – Enuncia a ronda do torneio em que está a ser disputado o jogo em questão.

Esta variável apresenta os valores distintos seguintes, estando organizada em qualificação e eliminatórias:

- 1st Round of Qualifying (1ª ronda de qualificação)
- 2nd Round of Qualifying (2ª ronda de qualificação)
- 3rd Round Qualifying (3ª ronda de qualificação)
- 3rd/4th Place Match (disputa pelo 3º/4º lugar)
- Round of 128 (1ª eliminatória)
- Round of 64 (2ª eliminatória)
- Round of 32 (3ª eliminatoria)
- Round of 16 (Oitavos de final)
- Quarter-Finals (Quartos de final)
- Semi-Finals (Semi finais)
- Finals (Finais)
- Olympic Bronze (disputa pela medalha de bronze)
- Round Robin (ronda em que os jogadores são separados em grupos e jogam todos contra todos, dentro do grupo, e quem ganhar mais jogos, passa à próxima ronda)

GameRank – Mostra o rank do jogador oponente, quanto menor, melhor é o jogador.

O ranking ATP funciona à base de pontos acumulados dos vários jogos dos quais um jogador sai vitorioso. Quantos mais jogos este ganhar, mas avança no torneio e mais pontos consegue

acumular, e por consequente, aumentar o seu rank. Esta acumulação de pontos difere de torneio para torneio (como mencionado anteriormente), dependendo do seu tamanho e prestígio. O cálculo do rank em si é feito através das 52 semanas em que decorrem torneios, sendo que os pontos têm essa mesma validade. Ou seja, um jogador que tenha ganho um torneio ATP 500 em 2022 chega a 2023 com esses 500 pontos, mas corre o risco de perdê-los se não voltar a ganhar o mesmo torneio, pelo que muitas vezes se ouve o termo 'defender os pontos'. No entanto, não são os pontos de todos os torneios jogados que são contabilizados, apenas os 19 melhores torneios do jogador são tomados em conta. Estes rankings são calculados e atualizados todas as semanas.

Oponent – Indica o primeiro e último nome do jogador oponente.

WL – Apresenta o resultado do jogo para o jogador principal, W se ganhou, L se perdeu.

Score – Revela o número de sets jogados e a pontuação de cada set.

2.3 Análise Exploratória na base de dados original

Numa fase inicial, foram feitas algumas análises à base de dados completa. Uma vez que já tinha sido desenvolvido um projeto com esta base de dados, os dados, as colunas e alguns problemas da base de dados já eram conhecidos (como a existência de jogos duplicados, entre outros). Contudo, uma vez que este projeto exigia um tratamento mais profundo dos dados, foi importante esta fase inicial do projeto, para entender quais eram os possíveis problemas que poderiam surgir durante a manipulação dos dados.

Foram feitas algumas visualizações para compreender qual o tipo de dados, quais as variáveis com valores omissos, quantas colunas é que existiam, qual a percentagem da base de dados que iria ser utilizada mais à frente, e uma série de outras questões que surgiram à medida que o trabalho foi realizado e que nos obrigaram a ter de voltar novamente a este passo inicial da compreensão da base de dados total.

Na base de dados inicial foi também importante conhecer quais os valores que existiam na variável Location, para ser possível avançar para o próximo passo e fazer uma filtragem apenas para os jogos disputados na China.

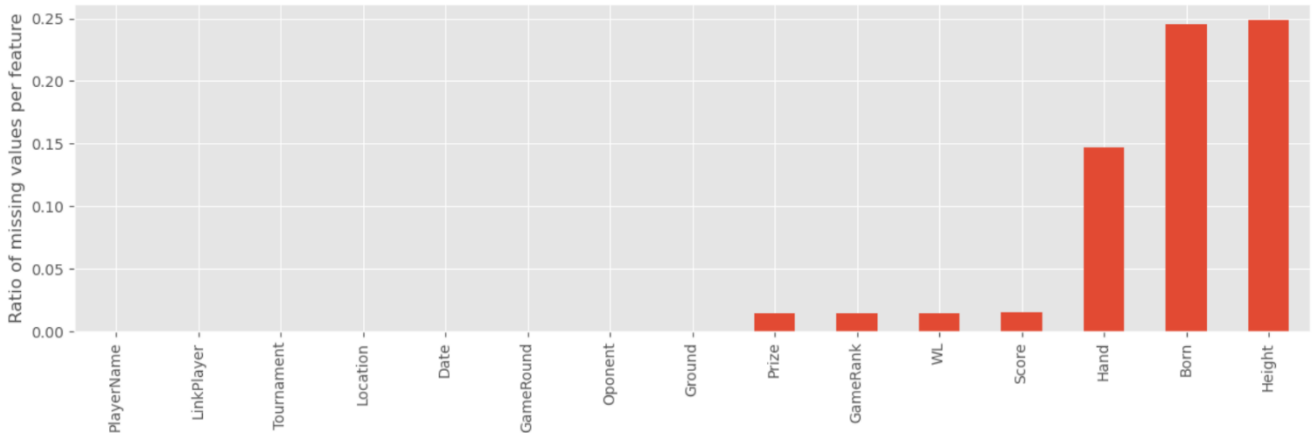

Figura 3 - Valores nulos em cada variável na base de dados original

2.4 Seleção dos torneios chineses

De modo a realizar as próximas etapas da metodologia CRISP-DM, e uma vez que este trabalho pretende analisar e prever sets utilizando uma base de dados de jogos da China, foram selecionados apenas os jogos que decorreram neste país.

Na base de dados original, existe a distinção entre China e Taipei, mas foi nos indicado pelo docente acompanhador que fossem considerados esses dois valores como sendo apenas China.

Posto isto, foram selecionadas as linhas nas quais o valor da variável ***location*** é China e Taiwan (na variável estes valores correspondiam a 'China', 'Taipei' e 'Taip'). Da base de dados original que continha 1308835 jogos, foram extraídos 26357 jogos que consideramos como tendo sido realizados na China, onde 23224 destes jogos tinham a variável igual a 'China' e 3133 tinham a variável igual a 'Taipei' e 'Taip'.

Estes jogos foram guardados num ficheiro .csv chamado de *atpchina.csv*, que foi depois usado para as seguintes fases do projeto.

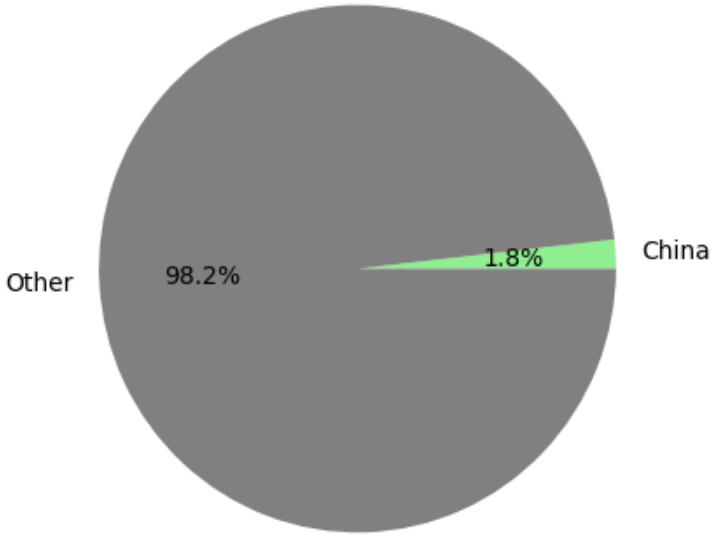

Figura 4 - Gráfico circular com a frequência relativa da China na base de dados total

2.5 Análise Exploratória sobre os dados da China

De seguida foram feitas algumas análises relativas aos dados da base de dados filtrada com jogos na China. Esta base de dados é constituída 26357 registos e 15 variáveis. Destas 15 variáveis, 7 continham valores omissos.

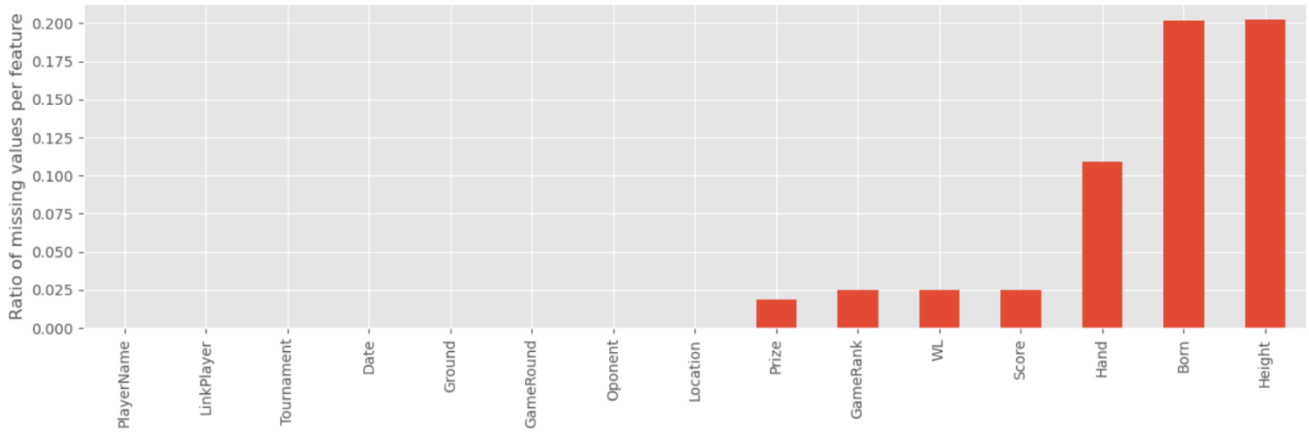

Figura 5 - Percentagem de valores omissos por variável

Variables	Number of nulls
PlayerName	0
Born	5314
Height	5341
Hand	2882
LinkPlayer	0
Tournament	0
Date	0
Ground	0
Prize	485
GameRound	0
GameRank	650
Oponent	0
WL	650
Score	652
Location	0

Tabela 1 - Número de valores nulos para cada variável

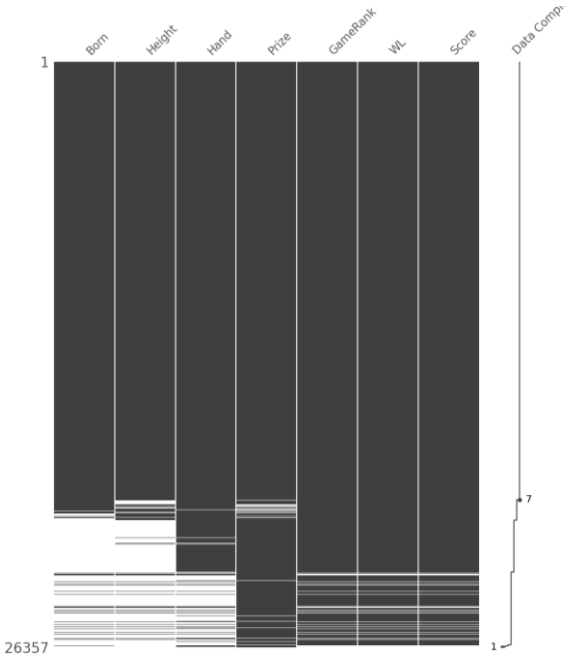

Figura 6 - Visualização das 7 variáveis com nulos e a sua posição na matriz

Observando a matriz da figura 6 é possível observar que muitos dos valores nulos coincidem às mesmas instâncias. Assim, se mais à frente for necessário apagar os valores nulos de algumas variáveis, é espectável que o número de nulos de outras variáveis também diminua.

A variável Score é essencial para a realização do trabalho, pelo que, não poderão existir valores nulos nesta variável. Portanto, como esta variável apresenta 652 observações omissas, optou-se por apagar estas observações.

Ainda, é possível observar os valores únicos para cada variável na tabela seguinte:

Variables	Number of unique values
Location	1
WL	2
Ground	3
Hand	7
GameRound	11
Height	22
Prize	55
Tournament	118
Date	380
Born	851
PlayerName	1663
LinkPlayer	1663
GameRank	1849
Oponent	2013
Score	2086

Tabela 2 - Número de valores únicos para cada variável

Para melhor compreensão e visualização, podemos observar mais facilmente estes valores no gráfico de barras seguinte.

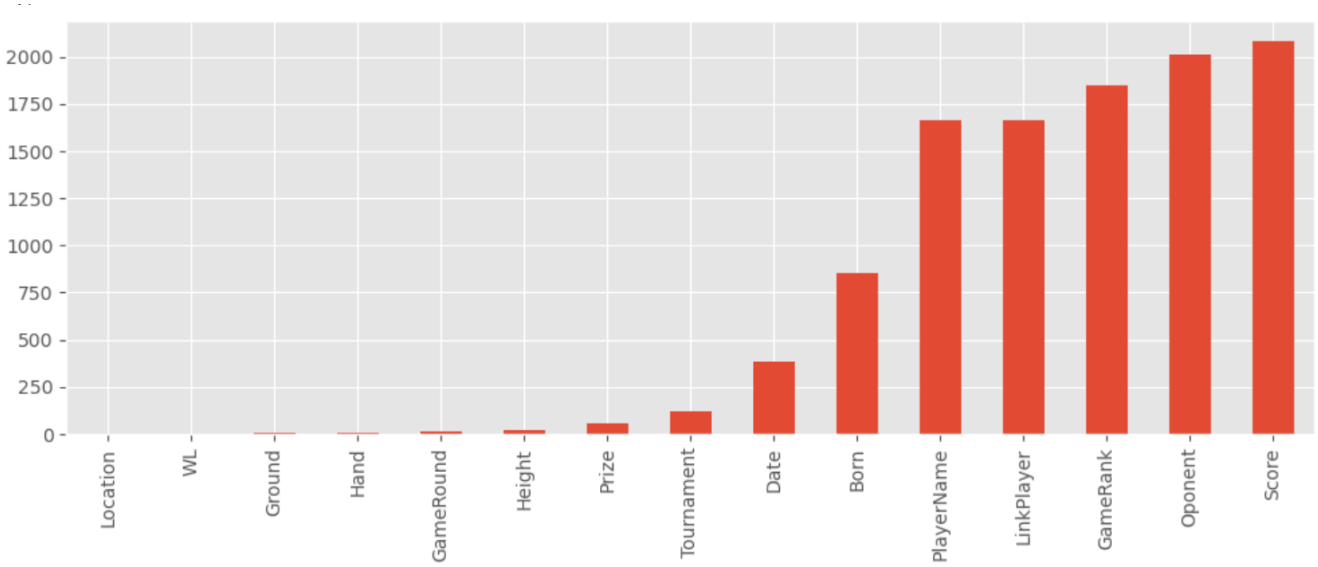

Figura 7 - Valores únicos para cada variável, por ordem crescente

Podemos também observar qual a distribuição de cada classe dentro de cada variável. Visto que que o número de variáveis é elevado, podemos fazer esta visualização apenas para as 4 variáveis que contêm o menor valor único (excluindo a variável Location).

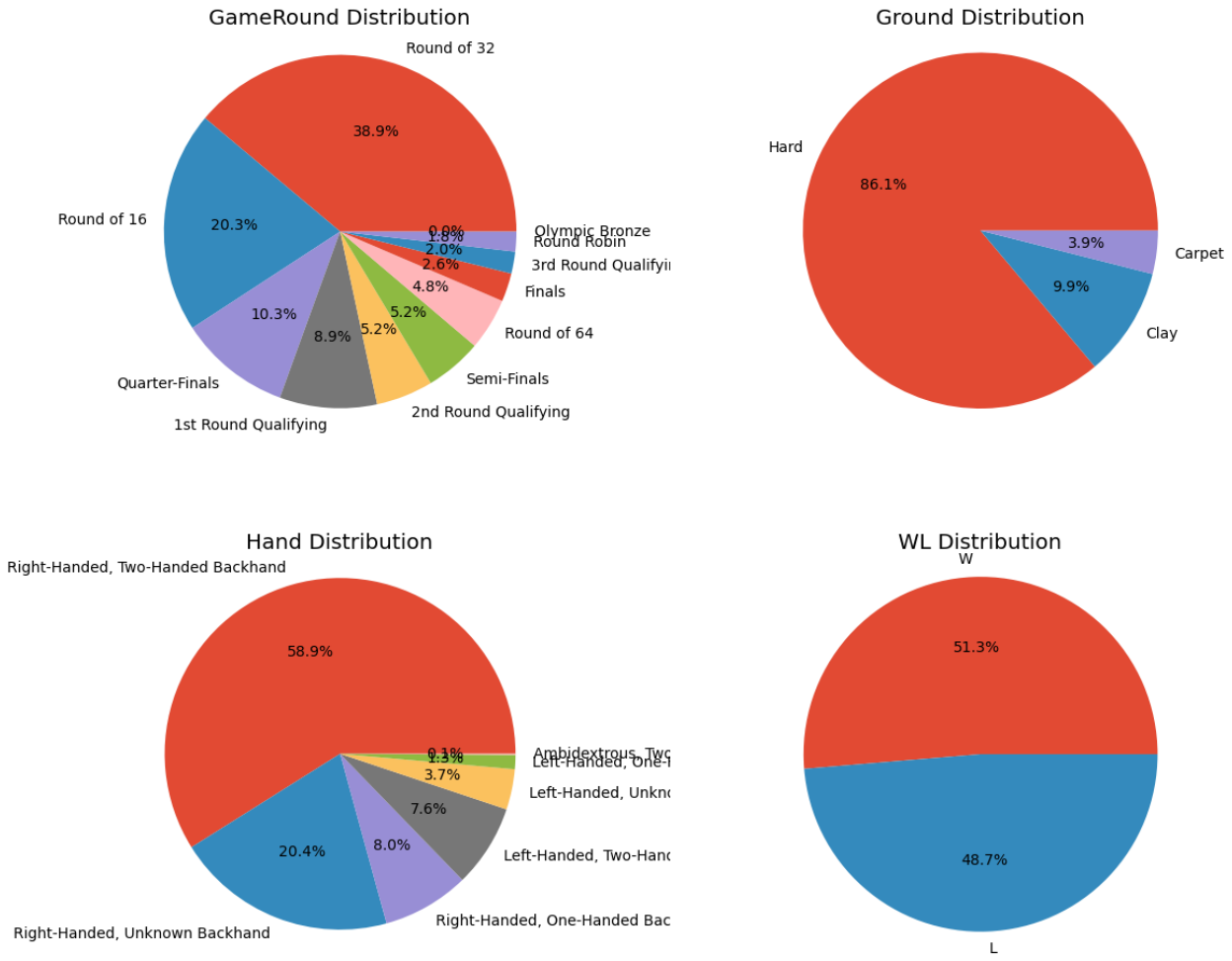

Figura 8 - Gráficos circulares das classes dentro de algumas variáveis

A partir dos gráficos circulares apresentados é possível observar que maior parte das variáveis contêm alguns problemas que serão abordados mais à frente quando for feita a limpeza.

No primeiro gráfico, que representa a distribuição das rondas do torneio em que os jogos são disputados, observamos que a maioria dos jogos (38.9%) ocorre na fase “Round of 32”. Em seguida, temos 20.3% dos jogos na fase de "Round of 16" e 10.3% nos “Quarter-Finals”. A fase de qualificação do “1st round” apresenta uma percentagem de 8.9%. As demais rondas possuem uma proporção muito pequena em relação ao total.

Em relação ao segundo gráfico, que mostra a distribuição dos tipos de terreno, podemos notar que a superfície predominante é o piso duro, representando 86.1% dos jogos. A superfície de terra batida corresponde a 9.9% dos jogos, enquanto a relva sintética representa apenas 3.9%. Não apresentando jogos em relva natural.

No terceiro gráfico, que analisa a preferência das mãos dos jogadores, observamos que a maioria (58.9%) é composta por jogadores destros com backhand de duas mãos. Em seguida, temos 20.4% de jogadores destros com backhand desconhecida. Os demais casos possuem proporções menores.

Por fim, no quarto gráfico, que representa a distribuição entre vitórias (win) e derrotas (lose), observamos que as vitórias ocorrem em 51.3% dos jogos, enquanto as derrotas correspondem a 48.7%.

Estas análises fornecem informações úteis para compreender a dinâmica dos jogos de ténis e auxiliar na elaboração de estratégias e previsões relacionadas com o número de sets necessários para concluir um jogo.

3. Data Preparation

Este capítulo corresponde à terceira fase da metodologia CRISP-DM. Esta foi a fase do trabalho que foi a mais demorada, uma vez que foram feitas alterações e limpezas às variáveis (incluindo, imputação de nulos, deteção de outliers, encoding, etc.) e também a criação de novas variáveis. Nesta fase vão ser explicadas todas as decisões tomadas em termos de limpeza, criação e eliminação de variáveis.

3.1 Tratamento dos Jogos

O primeiro passo a ser realizado foi a limpeza dos jogos. Como já foi escrito anteriormente, inicialmente tínhamos 26357 jogos.

Ao abrir a base de dados, identificamos que existiam jogos duplicados, jogos espelhados e jogos não espelhados.

Os jogos espelhados são os jogos em que os jogadores, a data e o resultado são os exatamente os mesmos, mas a ordem dos jogadores está invertida. Por outras palavras, os jogadores A e B enfrentam-se numa data, esse jogo será espelhado caso exista outro registo na base de dados, em que a data, competição e ronda do jogo são os mesmos, mas a ordem dos jogadores fica invertida; era possível identificar os jogos duplicados fazendo uso das colunas PlayerName, Oponent, Tournament e Date, onde todas as informações seriam iguais nos dois registos, à exceção nas colunas PlayerName e Oponent que iriam aparecer invertidas.

Os jogos não espelhados referem-se aos jogos em que apenas existe um único registo aonde as variáveis Date, GameRound e Tournament são iguais.

Os jogos duplicados são os jogos em que aparece duas vezes o mesmo registo. Ou seja, são os jogos onde, em todas as colunas, as informações são iguais. Isto é um erro da base de dados e como tal, estes tipos de jogos deverão ser eliminados.

Portanto, o primeiro passo que realizámos, foi a eliminação de observações duplicadas. Isto correspondeu à eliminação de 8 registos da base de dados.

3.2 Eliminação de outras observações

Durante o Data Understanding, identificámos três processos que deveriam ser rapidamente tratados, uma vez que não fazia sentido guardar jogos que não têm informação relevante:

-Na variável Score existiam observações com NA. Estas observações foram imediatamente eliminadas, uma vez que a variável target será construída a partir da Score; ou seja, eliminamos todas as observações com NA na coluna Score. Este processo correspondeu à eliminação de 450 observações.

-Ainda, na variável Score existiam observações que estavam da seguinte forma "(W/O)"; estas observações significam que ocorreu um "Walkover". Em ténis, um Walkover significa quando um jogador ganha um jogo sem ter de competir; isto pode acontecer, porque o adversário desiste antes de ocorrer o jogo. Como tal nestas situações, como nem chegou a existir jogo em si, decidiu-se também apagar estes jogos.

- Eliminação dos jogos amigáveis; também no Data Understanding foram identificados torneios amigáveis. Devido às diferenças de jogos em torneios amigáveis para torneios oficiais, decidimos eliminar todas as observações de jogos amigáveis.

3.3 Variável PlayerRank

Dado que existia uma variável que representava o ranking do jogador oponente (GameRank) e não existia nenhuma que representava o ranking do PlayerName, decidimos criar a coluna PlayerRank.

Assim, o PlayerRank, terá os rankings do jogador PlayerName.

Esta coluna teve de ser criada antes do tratamento dos jogos espelhados, uma vez que na maioria dos jogos esta coluna seria preenchida por esses jogos. Ou seja, se um jogo entre A e B for espelhado, então no registo em que A é PlayerName, o PlayerRank dessa observação será o GameRank do jogo em que B é o PlayerName uma vez que GameRank representa o ranking dos jogadores oponentes.

Com o processo acima descrito realizado, a coluna PlayerRank ficou com 232 observações omissas.

3.4 Eliminação dos Jogos Duplicados

Com a criação da coluna GameRank, já poderíamos proceder para a eliminação dos jogos espelhados.

O objetivo seria ter uma base de dados sem jogos espelhados, dentro dos quais o PlayerName era o vencedor do jogo; o objetivo então, seria ter todas as observações, onde na coluna WL, fosse "W" em todas as observações.

Então, criámos um algoritmo, em que caso existissem dois jogos onde Date, GameRound, Tournament e combinação dos nomes do PlayerName + Oponent fossem iguais, então esses jogos eram espelhados.

Chegamos à conclusão de que na nossa base de dados 23818 jogos eram espelhados e 1362 não eram espelhados. Como o objetivo era ter apenas observações onde WL=" W", então teríamos de verificar nos jogos que não eram espelhados, quantos existiam com esse requisito.

Após a verificação, concluímos que dos 1362 jogos, apenas 350 jogos tem a coluna WL=" L", como corresponde a menos de 1% de jogos, decidimos não espelhar esses 350 jogos.

Após estas operações criámos a nossa nova base de dados que contém 12921 jogos.

3.5 Data Transformation

Nesta segunda parte do Data Preparation, iremos agrupar colunas, criar colunas e imputar omissos.

Iremos então, criar variáveis relacionadas com índices físicos e de forma, que possam influenciar a duração de um jogo de ténis. Deste modo, variáveis como a idade, ranking ou forma recente têm um impacto direto na duração de um jogo, portanto, é nesta fase que vamos tentar explorar ao máximo essas situações.

3.5.1 Variáveis sobre Idade dos jogadores

Um dos índices físicos que influencia a duração de um jogo, é a idade dos jogadores.

A idade tem influência em vários aspetos no decorrer de um jogo, entre os quais:

- **Resistência**: Os jogadores mais jovens possuem normalmente mais resistência, isto permite com que possam adotar um nível de jogo mais intenso, com longas trocas de bola; também jogadores mais jovens tendem a recuperar mais rápido dos pontos jogados;

- **Velocidade e agilidade**: Jogadores mais jovens tendem a ser mais rápidos e ágeis, o que lhes permite movimentarem-se a uma velocidade maior pelo court, ou seja, jogadores mais jovens tem maior facilidade em cobrir o court que jogadores mais velhos;

- **Potência**: Jogadores mais jovens tendem a não ter tanta força nas "pancadas", ou seja, jogadores mais jovens tendem a meter menos força nos serviços;

- **Experiência e tática**: Com o passar dos anos, os jogadores vão ganhando experiência competitiva, isto permite-lhes ter mais conhecimento de jogo; ao acumularem mais experiência, os jogadores mais velhos vão saber ler as fraquezas dos jogadores adversários de forma mais rápida, saber gerir melhor os momentos dos jogos ou saber mudar estratégia durante um jogo;

- **Variedade de jogo**: jogadores mais velhos tendem a ter um maior arsenal de golpes, não é comum ver jogadores mais novos a fazer *amortis* ou *lobs*.

Portanto, para compararmos a idade dos jogadores, o objetivo seria então criar uma variável com a diferença de idades entre os jogadores. Para isso precisaríamos de criar duas colunas na base de dados:

-**BornDate**: Data de nascimento do PlayerName;

-**BornDate_Oponent**: Data de nascimento do Oponent.

Como tal, era necessário termos a data de nascimento dos jogadores e havia 2 formas de obter essas datas:

- Usando a técnica de Webscrapping;

- Ir à procura de bases de dados que tivessem essa informação.

Infelizmente, não foi possível usar a técnica de Webscrapping, mais em baixo será explicado de melhor forma o porquê de não termos executado essa técnica; por isso avançamos para a procura duma base de dados.

A base de dados permitiu preencher quase todas as observações, sendo que as datas que não conseguiram ser preenchidas, tiveram de ser preenchidas à mão. Portanto, nestas duas variáveis não foi necessário usar nenhum tipo de imputação.

Para auxiliar as duas variáveis em baixo que criámos, decidimos criar uma nova variável chamada Start_Date. A Start_Date vai à variável Date e fica com a primeira data (data de início do torneio).

Com as datas de nascimento de ambos os jogadores, foi possível também criar mais duas colunas referentes à idade dos jogadores no momento em que estavam a jogar o jogo, ou seja, criámos mais duas colunas chamadas:

- **AgePlayer**: Que consiste na subtração entre o Start_Date e o BornDate, esta coluna corresponde à idade do PlayerName no momento em que o jogo é jogado;

-**Age_Oponent**: Que consiste na subtração entre o Start_Date e o BornDate_Oponent, esta coluna corresponde à idade do Oponent no momento em que o jogo é jogado.

A partir daqui, só tínhamos de criar a diferença de idades entre os jogadores que corresponderia à subtração entre as observações das colunas Age_Player e Age_Oponent (nota, as observações resultantes desta subtração, deverão estar em módulo para não corrermos o risco de ter diferenças de idade negativas, uma vez que estamos a fazer a idade do PlayerName menos a idade do Oponent, sendo que na nossa base de dados, os PlayerName vencem os jogos todos).

Portanto, com base no que acima está explicado, criou-se a coluna Age_Difference, e posto isso, criou-se outra coluna com as diferenças de idade intervaladas (de forma, a controlar melhor a distribuição desta variável), sendo que consideramos os seguintes intervalos:

- Uma diferença de **0 a 5** anos, não tem impacto nenhum no decorrer do jogo;

- Uma diferença de **5 a 10** anos, poderá resultar em jogos mais longos, uma vez que o jogador mais velho deverá ter índices físicos parecidos ao mais novo;

- Uma diferença de **10 a 15** anos poderá resultar tanto em jogos mais curtos como mais longos;

- Uma diferença de **15 a 20** anos, deverá resultar em jogos mais curtos, onde aí o jogador mais novo deverá ter melhores índices físicos;

- Uma diferença superior a 20 anos, deverá resultar em jogos extremamente curtos.

Nota: esta coluna que é constituída por intervalos, tivemos de aplicar o método OneHotEncoder para ser possível usar em modelos.

3.5.2 WebScrapping

Como referido anteriormente, com o objetivo de preencher a variável BornDate, foi realizada uma tentativa de webscrapping (presente no notebook WebScrapping.ipyng).

Para isto, foi criado um ficheiro no formato .txt com os links dos jogadores e um ficheiro no formato .csv que continha duas colunas: LinkPlayer, que continha todos os valores únicos da variável LinkPlayer na base de dados da china e BornDate, uma coluna que estava vazia e

que iria ser preenchida pelo algoritmo. O algoritmo percorria cada link, que direcionava para o site ATP, e ia identificar e extrair a data de nascimento do jogador para preencher a coluna da tabela.

O algoritmo não funcionou, pois, passadas cerca de 50 iterações surgia um erro que indicava que tinha ocorrido um bloqueio por parte do website devido à quantidade excessiva de requests. Foi-nos recomendado que, para evitar esse problema, acrescentássemos um temporizador aleatório para que cada loop do código demorasse um tempo aleatório a correr. Mesmo depois da adição desse temporizador o código continuou a não funcionar por isso prosseguimos para a outra estratégia descrita.

3.5.3 Tratamento dos torneios

Um outro fator importante na duração de um jogo de ténis, é a qualidade dos torneios. Como já foi explicado em cima, nesta fase, só temos torneios que correspondem a jogos oficiais.

Mas dentro dos jogos oficiais, existem torneios mais importantes que outros, sendo que são nos torneios mais importantes ondem jogam os melhores jogadores.

Por exemplo, jogadores do nível do Rafael Nadal ou do Novak Djokovic não jogam Challengers.

Devido a este tipo de diferenças de qualidade de jogo e de qualidade dos jogadores nos diferentes torneios, decidimos agrupar os torneios, pela seguinte ordem:

- Torneios ITF;

- Torneios Challenger;

- Torneios ATP 250;

- Torneios ATP 500;

- Torneios ATP 1000, nesta categoria também adicionámos os Jogos Olímpicos uma vez que não fazia sentido criar uma categoria para tão poucos jogos.

Após a criação desta coluna com os torneios agrupados a que chamamos de TournamentType, o passo seguinte foi realizar o método OneHotEncoder sobre esta coluna. Assim, criámos a coluna Tournament_Encoded.

Sendo que esta coluna ficou com a seguinte distribuição:

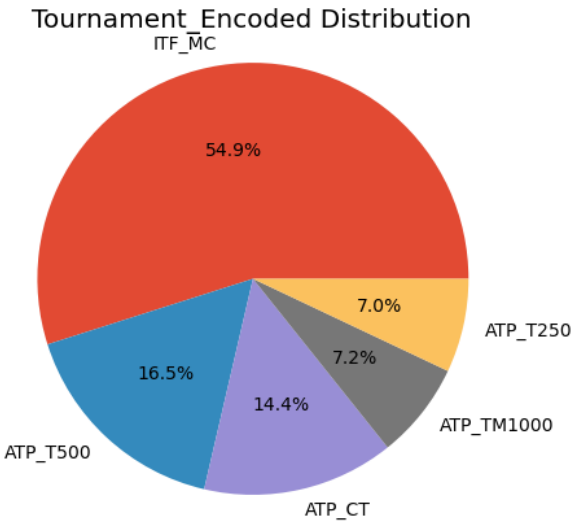

Figura 9 - Gráfico circular das classes da variável Tournament_Encoded

Nota: As codificações acima representam:

0 – Os torneios do tipo Challenger;

1 – Os torneios do tipo ATP250;

2 – Os torneios do tipo ATP500;

3 – Os torneios do tipo Masters100 (mais Jogos Olímpicos);

4- Os torneios do tipo ITF.

3.5.4 Estações do ano

Um dos fatores que não são relacionados com o jogador, que influenciam a duração do jogo, é a temperatura e as condições meteorológicas que decorrem durante o jogo.

Em baixo, estão presentes alguns fatores nos quais as estações do ano influenciam os jogos:

- **Temperatura**: A temperatura tem um impacto significativo no decorrer dos jogos. Durante o verão, quando as temperaturas são mais elevadas, as bolas tendem a saltar mais e os courts ficam mais rápidos de se movimentar (só se aplica quando o jogo é jogado em piso duro); sendo que jogos em temperaturas mais elevadas, favorecem jogos mais intensos e com mais ofensivos; em temperaturas mais baixas, os courts ficam mais pesados e as bolas tendem a saltar menos, sendo que, este tipo de temperaturas favorecem jogadores com um tipo de jogo mais defensivo, resultando em trocas de bola mais longas;

- **Humidade**: Em condições húmidas, a bola tende a ficar mais pesada e os courts podem ficar mais lentos, sendo que isso irá dificultar a velocidade das bolas. Logo condições mais húmidas, tornam o jogo mais lento, favorecendo os jogadores com tipo de jogo mais defensivo e mais pacientes, obviamente, que estes fatores resultam em jogos com trocas de bolas mais longas;

- **Vento**: Em temperaturas mais baixas, costuma existir mais vento. Sendo que o vento pode afetar a trajetória da bola, dificultando o controlo e precisão dos golpes;

- **Luz Solar**: A intensidade da luz solar também influencia os jogos. Sendo que obviamente no verão, o jogo tem maior intensidade solar. Em condições de luz solar intensa, a sombra pode criar contrastes fortes de visibilidade em todo o court, dificultando a visibilidade da bola.

De notar ainda, que quanto mais elevadas as temperaturas, maior o cansaço acumulado nos jogadores, sendo que isso resulta em trocas de bola mais curta, visto que assim os jogadores vão descansado mais durante os pontos jogados.

Como tal, nesta fase, criou-se a coluna Seasons_Encoded que terá quatro valores diferentes: primavera, verão, outono e inverno. Sendo que como aplicamos o OneHotEncoder, assumirá valores: 0,1,2,3.

Como também é possível notar em baixo, podemos concluir que esta variável ficou bem distribuída e não deverá prejudicar o modelo:

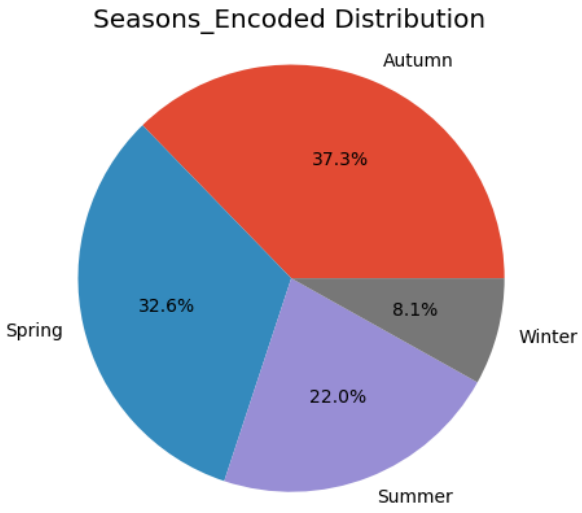

Figura 10 - Gráfico circular das classes da variável Seasons_Encoded

3.5.5 GameRound

Em cada torneio, temos rondas completamente diferentes, sendo que, o nível de jogo vai variando conforme as rondas onde são jogados.

No Data Understanding, já foi explicado quais as diferentes rondas que existem, nesta fase, vamos tentar agrupar as rondas conforme a dificuldade do jogo.

Agrupámos as rondas da seguinte maneira:

-Todas as rondas de qualificação deveriam ser agrupadas como 'Qualifying': na fase de qualificação, a pressão de ganhar para se qualificar para o quadro principal é muito maior. Sendo que certos jogadores, podem estar motivados para dar o seu melhor. Também nesta fase, não existe uma grande diferença entre a qualidade dos jogadores;

-As rondas iniciais do torneio, 'Round of 64', 'Round of 32', 'Round of 16' e 'Quarter-Finals' deveriam ser agrupadas como 'Early Stages': Nas primeiras rondas de um quadro principal, as qualidades dos jogadores variam imenso. Nesta fase, podemos ter alguns dos melhores jogadores do mundo a enfrentar jogadores muito mal classificados no ranking ATP, sendo que este tipo de jogos, resultam em jogos mais simples e muito mais rápidos;

-E por fim, as seguintes rondas: 'Semi-Finals', 'Finals' e 'Olympic Bronze' deveriam ser agrupadas como 'Final Stages': Aqui a qualidade dos jogadores já deve ser mais comparável e o nível de competição aqui é extremamente elevado, os jogadores nesta fase já jogam a lutar por títulos, sendo que isto dá mais motivação aos próprios jogadores.

Decidiu-se deixar o Round Robin numa única categoria, uma vez que esta é uma ronda típica dos ATP Finals (torneios jogados tipicamente no final do ano, onde todos jogam contra todos, uma espécie de fase de grupos). Sendo que nos ATP Finals, só jogam os jogadores mais bem posicionados no ranking ATP, ou seja, os jogadores são de qualidade superior e os jogos são extremamente renhidos.

3.5.6 Ranking

Os rankings também fazem a diferença num resultado de ténis. Uma diferença grande entre o ranking de dois jogadores deverá traduzir num jogo rápido.

Os rankings podem influenciar um jogo, desde o posicionamento no sorteio até ao tipo de jogo, em baixo, temos algumas formas de como o ranking pode influenciar um jogo de ténis de forma direta e indireta:

- **Posicionamento no sorteio**: Esta é uma forma em que o ranking influencia de modo direto um jogo. Os rankings determinam o posicionamento dos jogadores no sorteio de um torneio.

Por norma, jogadores com ranking elevado não calham contra outros jogadores de ranking elevado nas primeiras rondas, uma vez que durante os sorteios existem regras especificas para os melhores jogadores não jogarem entre si nas primeiras rondas;

- **Confiança e mentalidade**: Esta é uma das formas em que o ranking influencia um jogo de forma indireta. Jogadores com ranking mais elevado geralmente entram num jogo com uma maior confiança e mentalidade mais positiva.

Como já foi referido em cima, antes da eliminação dos jogos espelhados, desenvolvemos uma coluna chamada PlayerRank com os rankings do PlayerName.

O problema é que dentro da variável PlayerRank e OponentRank (era a antiga variável GameRank) para além das observações que tínhamos com omisso, tínhamos ainda observações com valor a 0.

Para imputar as observações omissas nestas variáveis, escolhemos duas alternativas para tal:

-Pela média, esta hipótese foi rapidamente recusada, uma vez que desta forma, caso um jogador tivesse um ano em que tivesse jogado muito bem, esse ano iria influenciar da mesma forma que todos os restantes anos influenciariam no cálculo do ranking. Por exemplo, Guillermo Coria que em 2004 foi à final do Open da França, atingiu nesse mesmo ano a posição número 3, mas passado um ano já nem no Top 20 se encontrava. Podíamos também usar o exemplo atual do Rafael Nadal, que foi durante 15 anos um jogador que se encontrou no top 10, e hoje, por exemplo, é o número 15 no ranking;

- Pela mediana, esta hipótese pareceu-nos a melhor uma vez que representaria de melhor forma o nível que um jogador se apresentou durante a sua carreira.

Então a decisão, foi dado um nome de um jogador, caso o seu ranking estivesse uma observação NA, substituir pela mediana do seu ranking.

Após estas imputações, ficámos com 96 observações omissas no PlayerRank e 632 no OponentRank.

Em relação ao PlayerRank, decidimos imputar os restantes valores à mão pela sua melhor posição encontrada. Em relação aos omissos do OponentRank, como os jogadores oponentes eram aqueles que perdiam, e muitos só tinham um jogo tanto na base de dados da China como na base de dados ATP inteira, decidimos imputar esses valores por 9999.

A partir daqui a única forma de podermos usar os rankings como variável preditora, de forma a não manipular os modelos, era criando intervalos.

Então, o que realizámos, foi criar uma coluna chamada de Ranking_Diff que consistia nas diferenças de ranking entre os jogadores que estão no PlayerName e os jogadores oponentes. Estes valores estiveram em módulo para não termos diferenças negativas.

A seguir criámos intervalos entre as diferenças de ranking para tentar retirar o peso das imputações que realizamos. Os intervalos foram os seguintes:

- **[0,50]** - Uma diferença pequena de ranking não deverá ter um grande impacto na duração do jogo. Assim, a diferença é demasiado pequena para podermos dizer que o jogo vai ser de pequena duração; sendo que à partida pequenas diferenças de ranking entre os jogadores, deverão resultar em jogos renhidos;

- **[50,100]** - Uma diferença moderada de ranking, deverá resultar em jogos renhidos, apesar de não tão renhidos como o intervalo em cima

- **[100,200]** - Esta é uma diferença mais elevada de ranking. O jogador com menor ranking deverá ter maior superioridade, mas não é suficiente para automaticamente dizermos que o jogo vai ser de curta duração.

- **[200,500]** - Aqui, esta diferença de ranking é possível dizer que o jogador com menos ranking já deverá ter mais 'skill', poderá ser possível dizer que os jogos serão de curta duração;

- A partir de **500**, aqui esta diferença de ranking é estrondosa, podendo dizer que os jogos serão de curta duração.

Sendo que, a variável ficou distribuída da seguinte forma:

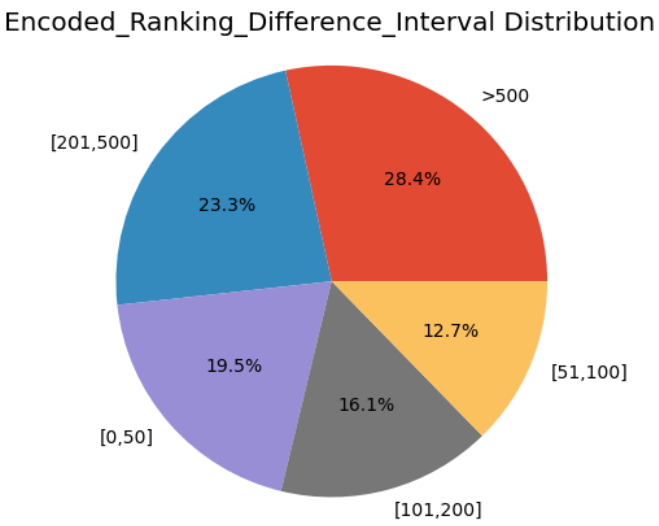

Figura 11 - Gráfico circular das classes da variável Encoded_Ranking_Difference_Interval

Posto isto, codificámos esses intervalos, para termos uma variável categórica de forma que fosse possível usar nos modelos (usamos o OneHotEncoder para tal).

3.5.7 Altura

A altura de um jogador faz com que um jogador alto tenha outras capacidades que um mais baixo não tenha, e vice-versa. Algumas diferenças são mais visíveis que outras, sendo que, entre as mais visíveis estão:

- **O serviço**: Jogadores mais altos tem maior vantagem neste aspeto de jogo, uma que tem mais força e maior alcance na bola;

- **Agilidade**: Jogadores mais altos tem menor agilidade que jogadores mais baixos;

- **Jogo de rede** - Jogadores mais altos geralmente têm uma vantagem nas jogadas de rede, uma vez que a sua altura permite intercetar bolas de forma mais fácil.

Em relação à altura, também tivemos o mesmo problema que nos rankings. Primeiro tínhamos de criar uma coluna que representasse a altura dos jogadores oponentes (uma vez que a coluna Height que já existia, só tinha a altura dos PlayerName). Após a criação da coluna (Oponent_Height que representava as alturas dos oponentes), reparamos que existiam alguns omissos em ambas as colunas da altura.

Como tal, tentámos realizar uma imputação por HotDeck usando o método KNN; infelizmente esta alteração não se revelou útil, uma vez que imputava em todos os omissos por 182. Independentemente do número de vizinhos que mais próximos que usávamos, com k=3 e acabámos com k=10. Isto acontecia, uma vez que este algoritmo não conseguia encontrar nenhum padrão entre os jogos.

Então, como a altura tem uma influência enorme no tipo de jogo, substituímos os omissos pela média de alturas dos jogadores do mesmo continente. Ou seja, jogadores nascidos na China e que tinham na altura omissos, seriam imputados pela média de alturas dos jogadores asiáticos.

A altura também se decidiu usar pela diferença entre a altura dos jogadores, uma vez que só dessa forma, era possível comparar, se existia algum padrão dentro da nossa base de dados entre jogos longos ou jogos curtos e a diferença entre as alturas de dois jogadores.

Portanto, definimos os seguintes intervalos de diferenças de alturas entre os jogadores:

- **[0,5]**: Nestes casos, a diferença de altura não é percetível, sendo que a diferença entre estilo de jogo dos jogadores, não deverá ser influenciada pela altura;

- **[6,10]**;

- **[11,15]**;

- **[16,20]**;

- **>20**: Nestes casos, diferenças de 20 centímetros, deverão resultar em diferenças de tipo de jogo entre os jogadores extremas. O jogador mais alto deverá apostar muito no seu serviço e em um tipo de mais lento, enquanto, o jogador mais pequeno deverá apostar num tipo mais rápido, uma vez que os jogadores são mais lentos a percorrer o court.

Portanto, o objetivo de usar essa variável era tentar encontrar um padrão entre diferentes estilos de jogo devido à altura, e se corresponde a jogos mais renhidos ou não.

3.5.8 Prize

Jogos com maior prémio monetário, estão associados também a jogos de maior qualidade. Por exemplo, um prémio de jogo correspondente a um torneio Masters 1000 deverá ser superior a um prémio de jogo correspondente a um torneio de Challenger.

Apesar de já todos os prémios estarem sobre dólares americanos, o grande problema desta coluna é que tínhamos de atualizar a maioria dos prémios de jogo, para a taxa de inflação atual.

Para isso produzimos uma base de dados, onde dado um ano, faria a atualização dos dólares daquele ano para o atual. Apesar de só termos jogos até 2022, uma vez que o trabalho está a ser realizado em 2023, atualizamos todos os prémios de jogo para as taxas de inflação de 2023.

Após isso, decidimos intervalar os prémios de jogo para comparar se o modelo trabalhava melhor com os prémios de jogo intervalados, ou com os prémios de jogo sem estar intervalados.

Os intervalos que criamos, foram entre: (0,20 000), (20 00, 25 000), (25 000, 50 000), (50 000, 100 000), (100 000, 1000000) e > 1000000.

3.5.9 Tiebreak e Points_Diff

Como o nosso modelo é para ser aplicado no âmbito das apostas desportivas, então, decidimos que o nosso modelo ia ser aplicado em 2 fases:

- Prever o número de sets antes do jogo começar;

- Prever o número de sets, depois do primeiro set acabar.

Com base no último tipo de modelos, decidimos criar mais duas variáveis que são apenas construídas quando o primeiro set acaba:

- **TieBreak**: esta variável vai buscar o resultado do primeiro set, e caso o primeiro set acabe com TieBreak , a observação dessa coluna tem 1; caso não tenha, terá 0;

- **Points_Diff**: esta variável serve para ver se a diferença no primeiro set foi grande, caso o primeiro set tenha acabado em 6-0, 6-1 ou 6-2 (ou vice-versa, 0-6, 1-6 ou 2-6), então essa observação terá 1 na coluna Points_Diff; caso não tenha, terá 0.

A ideia por trás destas variáveis, é que caso o primeiro set tenha ido a Tiebreak, então à partida, o segundo set também será renhido; e a mesma ideia se aplica para o Points_Diff, porque caso o primeiro set não tenha sido renhido, à partida, o segundo também não deverá ser.

3.6 Criação da variável Target

Neste subcapítulo iremos abordar sobre a criação da variável target, à qual apelidamos de "Sets".

A variável Sets teve de ser construída à base da variável Score, uma vez que era dentro de Score que estavam os resultados do jogo, e que só a partir dos resultados do jogo é que podíamos contar o número de sets no jogo.

Então tivemos que desformatar algumas observações dentro de Score, desde espaços a mais entre os jogos, ou alguns jogos que estavam definidos como "4-6, 4-6", e outros que tinham o resultado assim: "4:6,4:6".

Após essas alterações todas, ficamos com os seguintes números de sets:

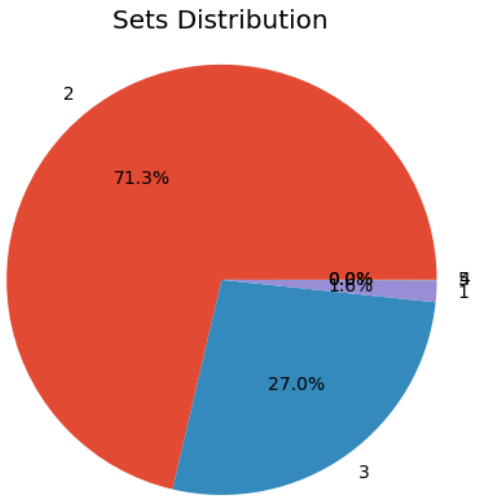

Figura 12 - Gráfico circular das classes da variável Sets

Ou seja, na nossa base de dados, a distribuição da nossa variável Sets, era a seguinte:

- 210 jogos acabavam no primeiro set;

- 9217 jogos acabavam no segundo set;

- 3491 jogos acabavam no terceiro set;

- 1 jogo acabava no quarto set;

- 2 jogos acabavam no quinto set.

Como atualmente, apenas só nos torneios de Grand Slam (Open de França, Wimbledon, Open da Austrália e Open dos Estados Unidos da América) existem jogos que podem ser jogados até aos quatro e cinco sets, decidimos apagar da base de dados esses três jogos.

Mais tarde, também decidimos apagar os jogos que acabam no primeiro set, pelas seguintes razões:

- Falta de representatividade dos dados, estes jogos que acabam no primeiro set, apenas representam 1.6% dos dados todos, pelo que seria um erro, com tão poucos dados tentar prever jogos que acabam no primeiro set;

- Redução de Ruído: a verdade é que não é normal um jogo acabar no primeiro set, os jogos só acabam no primeiro set, quando há jogadores que se são obrigados a retirar-se durante a realização do mesmo; como tal, ao excluir estes jogos, estamos a reduzir possível ruído associado a estes jogos e a melhorar a capacidade do modelo em identificar padrões relevantes para prever sets em jogos completos.

3.7 Variáveis que não usamos

Antes de avançarmos para o penúltimo passo do CRISP-DM, devemos explicar que existiram duas variáveis nas quais não fizemos alteração nenhuma e que durante o Data Understanding percebemos que não as iriamos usar, as duas variáveis são:

- **Ground**;

- **Hand**;

3.7.1 Ground

Existem razões válidas para não utilizar a variável "Ground" nos modelos. Primeiramente, a distribuição é desequilibrada, ou seja, 86.1% dos valores desta variável correspondem a um único tipo de terreno, o piso duro. Esta falta de diversidade na distribuição dos dados relacionados ao tipo de terreno pode afetar negativamente a capacidade de os modelos realizarem previsões precisas e generalizáveis.

Se houver uma falta de representação adequada de diferentes tipos de terreno na base de dados e os modelos forem treinados com base nesses dados não balanceados, existe o risco de que os modelos se ajustem excessivamente ao tipo de terreno dominante, perdendo a capacidade de generalizar para outros cenários e superfícies.

Mais, embora o tipo de terreno possa influenciar o estilo de jogo e o desempenho dos jogadores de ténis, o seu impacto direto na duração dos sets pode ser menos significativo em comparação com outras variáveis disponíveis.

Ao remover a variável "Ground" dos modelos, podemos reduzir a dimensionalidade do conjunto de dados, simplificando assim a complexidade dos modelos, facilitando a sua interpretação. Focar em variáveis mais impactantes e diretamente relacionadas ao número de sets pode melhorar a precisão e a eficiência dos modelos.

3.7.2 Hand

Primeiro, a variável "Hand" não está diretamente relacionada à duração dos sets num jogo de ténis. Esta indica apenas a mão dominante do jogador, não influenciando diretamente a quantidade de sets necessários para concluir uma partida. A mão dominante pode ser relevante para outros aspetos do jogo, como estratégia e preferência por determinados golpes, mas não tem impacto direto nos sets.

Segundo, a informação sobre a mão dominante é estável ao longo da carreira de um jogador. Como a mão dominante é uma característica pessoal que não muda com frequência, pode não fornecer informações adicionais relevantes para prever o número de sets em cada jogo. A sua inclusão nos modelos de previsão poderia não contribuir de forma significativa para os resultados.

Terceiro, não há uma relação causal direta entre a mão dominante do jogador e a duração dos sets. Embora possa haver diferenças sutis entre jogadores destros e canhotos em termos de estilo de jogo, não há evidências conclusivas de que a mão dominante influencie diretamente a quantidade de sets em uma partida.

Ao analisar as variáveis disponíveis na base de dados, optámos por não utilizar a informação sobre a mão dominante dos jogadores na previsão do número de sets. Concentrámos a nossa atenção em variáveis mais relevantes, como ranking dos jogadores e histórico de jogos anteriores, que possuem uma relação mais direta e significativa com a duração dos sets em jogos de ténis profissional. Essas variáveis oferecem uma base mais sólida para a previsão dos sets e permitem obter resultados mais precisos e confiáveis.

3.8 Base de dados original

Com o objetivo de criar mais variáveis, foi necessária uma limpeza da base de dados total. A limpeza da base de dados na sua totalidade permitiu a criação de outro tipo de variáveis que antes não eram possíveis de criar. Estas variáveis são variáveis que têm por base jogos anteriores de um jogador para tentar encontrar padrões que alguns jogadores possam ter, tendo em conta jogos do passado.

3.8.1 Limpeza

Uma vez que a base de dados total irá servir como um histórico de jogos, para facilitar a manipulação e a limpeza da mesma, foram retiradas as variáveis 'Hand', 'LinkPlayer', 'Height', 'Born' e 'Prize'. Foram também eliminados os valores nulos da variável 'Score', tal como tinha sido feito na base de dados da china. Posteriormente a variável 'Score' teve de ser limpa, para ser possível criar a variável 'Sets' também nesta base de dados. A variável 'Date' também sofreu algumas alterações, onde foi retirada a variável 'StartDate', que é necessária para criar as futuras variáveis.

Com a base de dados completa suficientemente limpa para o objetivo pretendido, foram então criadas as seguintes variáveis:

- 'PlayerName_RecentPerformance_Encoded'
- 'Oponent_RecentPerformance_Encoded'
- 'H2H_Encoded'
- 'TimeDifference_Interval_Encoded'

3.8.2 Criação das variáveis

'PlayerName_RecentPerformance_Encoded' e 'Oponent_RecentPerformance_Encoded'

As variáveis criadas que mostram e descrevem a performance recente de um jogador foram criadas tendo em conta os seus últimos 5 jogos. O valor da performance recente é um valor numérico, calculado a partir da média de sets dos últimos 5 jogos de um jogador, que foi arredondada para valores de 2 e 3. Assim, o esperado é que quanto maior for esse valor, maior vai ser o número de sets do próximo jogo (do jogo que queremos prever). No final essa variável foi codificada para valores de 0 e 1.

'H2H_Encoded'

O nome desta variável surge da expressão Head-to-Head, que é essencialmente o significado desta variável. O nosso objetivo com a criação desta variável é analisar todos os jogos em que os 2 jogadores se defrontaram, ou seja, todos os jogos que o PlayerName jogou contra o Oponent. Esta variável é também uma média do número de sets dos jogos entre esses mesmos jogadores, que depois é arredondada para 2 e 3 e mais à frente convertida para 0 e 1. Quando era a primeira vez que os jogadores se estavam a defrontar o valor inserido foi 0.

'TimeDifference_Interval_Encoded'

Esta variável compara o número de dias desde o último jogo do PlayerName e o número de dias desde o último jogo do Oponent. É calculada usando a data do jogo atual e a data do último jogo jogado. Foi calculado um valor para cada um dos jogadores e depois fez-se uma diferença entre os dois valores, subtraindo o valor do Oponent ao do PlayerName. Para estes valores foram criados os seguintes intervalos: [0,14]; [15,30]; [31,60]; >60. Para finalizar, estes intervalos foram então codificados desta maneira: [0,14] – 1; [15,30] - 2; [31,60] - 3; >60 – 0.

4. Modeling

Nesta fase do trabalho irão ser abordados os procedimentos para corrigir o desequilíbrio presente na variável alvo. Além disso, será discutida a seleção dos melhores modelos e sua implementação.

Os modelos iniciais foram treinados e testados com uma seleção de variáveis que foram consideradas interessantes e importantes para a previsão do número de sets.

Estas variáveis são:

- 'Seasons_Encoded'
- 'GameRoundFases_Encoded',
- 'Encoded_Age_Difference_Interval'
- 'Encoded_Ranking_Difference_Interval'
- 'Encoded_Height_Difference_Interval'
- 'TieBreak'
- 'Tournament_Encoded'
- 'PlayerName_RecentPerformance_Encoded'
- 'Oponent_RecentPerformance_Encoded'
- 'H2H_Encoded'
- 'TimeDifference_Interval_Encoded'
- 'Present_Prize'
- 'Prize_Intervals_Encoded'
- 'PointDiff'
- 'Age_Difference'

Muitas destas variáveis não vão fazer parte do modelo final, mas por agora foram consideradas importantes para fazer testes antes de selecionar o modelo ideal.

4.1 Desequilíbrio da variável alvo

Antes de iniciar o processo de Modeling, observou se que a variável alvo 'Sets' estava muito desequilibrada, onde aproximadamente 72.5% das observações correspondiam a 2 sets e aproximadamente 27.4% das observações correspondiam a 3 sets.

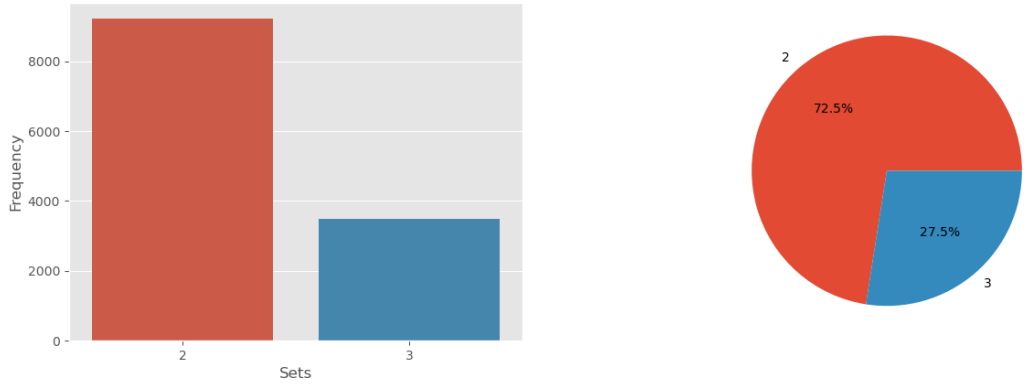

Figura 13 - Visualização gráfica dos valores únicos da variável alvo

Para resolver este problema surgiram três soluções diferentes: Oversampling, Undersampling e Stratified Sampling. Estas três soluções foram aplicadas sempre aos mesmos três modelos, de forma a ser possível compará-las entre si. Estes modelos utilizados foram: o modelo Random Forest, o modelo KNN e o modelo Naïve Bayes.

4.1.1 Oversampling

O Oversampling é uma técnica utilizada para lidar com o desequilíbrio de classes em conjuntos de dados. O que esta técnica faz é aumentar o número de exemplares da classe minoritária, no nosso caso seria a classe 3, e fazer com que ambas as classes tenham o mesmo número de exemplares.

Existem vários métodos que permitem fazer oversampling, mas o utilizado foi o SMOTE (Synthetic Minority Oversampling Technique). O que o SMOTE faz é criar exemplares da classe minoritária através de interpolação entre os pontos mais próximos.

Após ser aplicado este método na base de dados de treino, foram treinados 3 modelos para observar os resultados. Na tabela seguinte conseguimos observar os valores de accuracy e AUC de cada modelo:

	Accuracy	AUC
Random Forest	0.6310	0.57
KNN	0.6884	0.54
Naïve Bayes	0.7144	0.57

Tabela 3 - Accuracy e AUC dos modelos com Oversampling

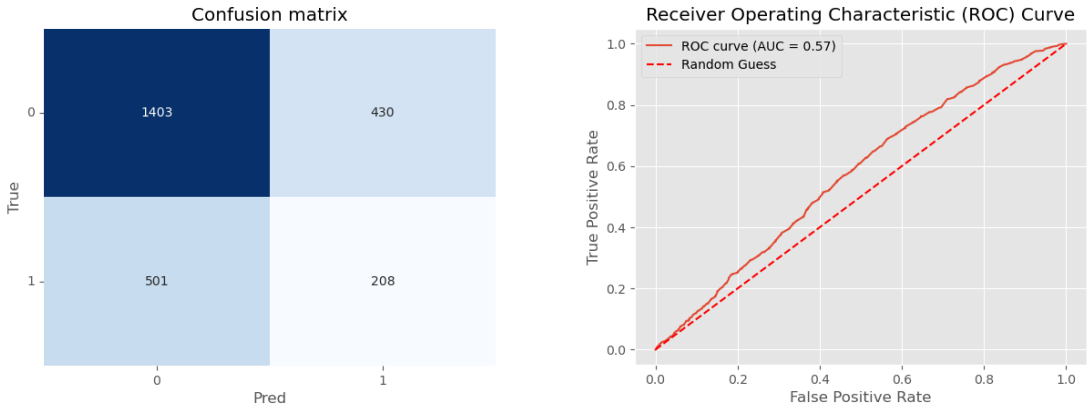

Figura 14 - Matriz de confusão e curva ROC do Random Forest

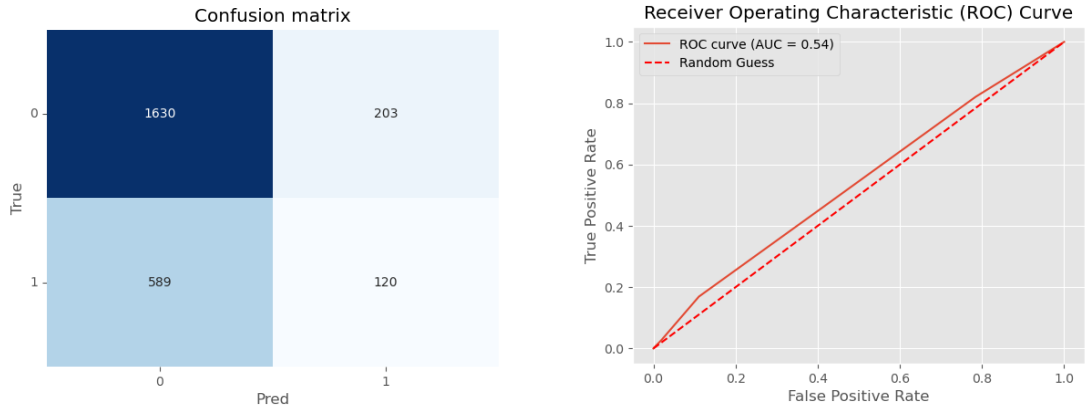

Figura 15 - Matriz de confusão e curva ROC do KNN

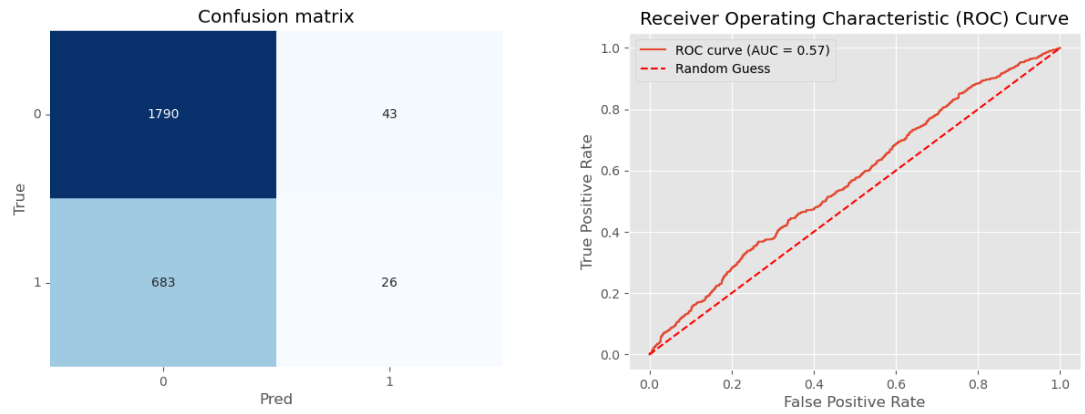

Figura 16 - Matriz de confusão e curva ROC do Naïve Bayes

4.1.2 Undersampling

Enquanto o *oversampling* aumenta a quantidade de exemplos da classe minoritária, o *undersampling* faz o contrário: ele diminui a quantidade de exemplos da classe maioritária.

Existem também vários métodos que permitem fazer *undersampling*, mas o utilizado foi o Random Undersampling, que apaga valores aleatórios da classe maioritária.

Após ser aplicado este método na base de dados de treino, foram treinados os mesmos 3 modelos para observar os resultados. Na tabela seguinte conseguimos observar os valores de *accuracy* e AUC de cada modelo:

	Accuracy	AUC
Random Forest	0.5496	0.58
KNN	0.6884	0.54
Naïve Bayes	0.7144	0.57

Tabela 4 - Accuracy e AUC dos modelos com Undersampling

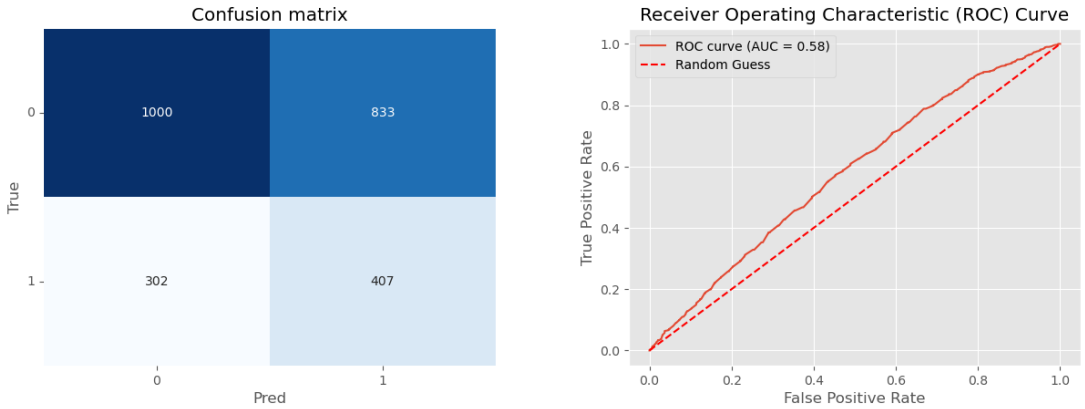

Figura 17 - Matriz de confusão e curva ROC do Random Forest

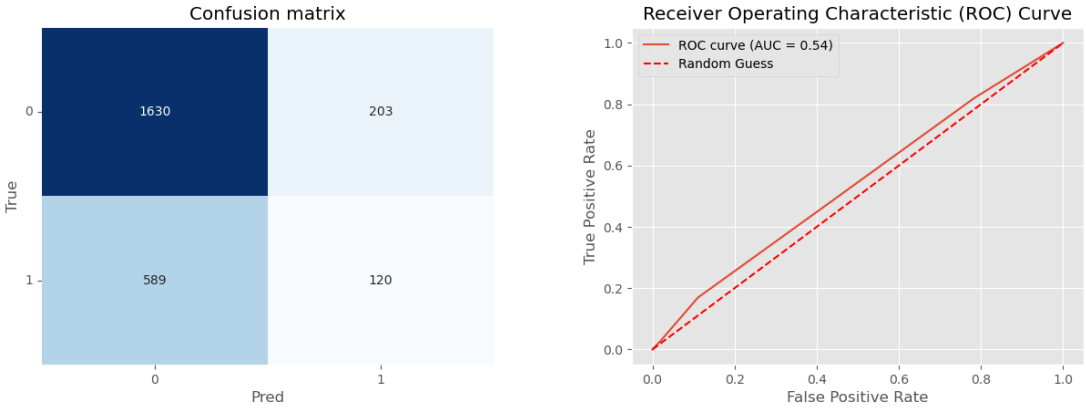

Figura 18 - Matriz de confusão e curva ROC do KNN

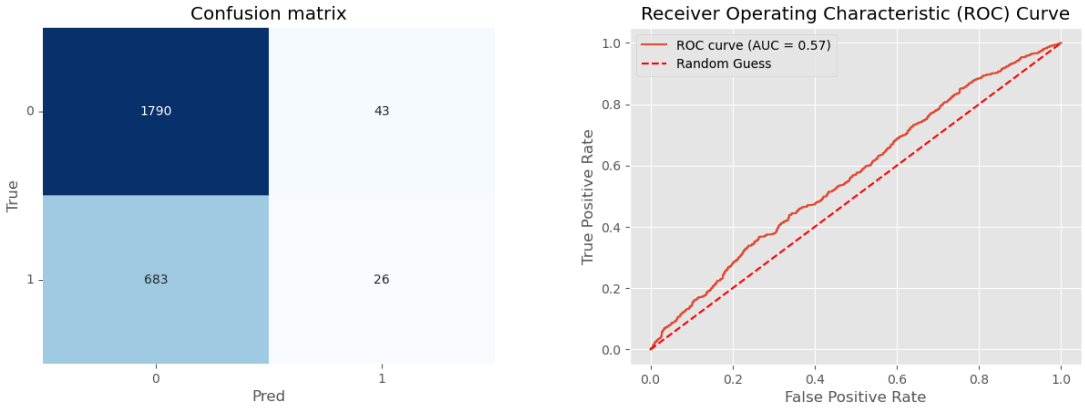

Figura 19 - Matriz de confusão e curva ROC do Naïve Bayes

4.1.3 Stratified Sampling

O Stratified Sampling é uma técnica estatística utilizada na seleção de uma amostra representativa de uma população. Com este método, a população é dividida em grupos com características semelhantes, sendo selecionada uma amostra de cada grupo. Isso garante que a amostra final reflita adequadamente a distribuição das características importantes presentes na população, reduzindo o viés e aumentando a precisão das previsões. Após ser aplicado este método na base de dados de treino, foram treinados os mesmos 3 modelos para observar os resultados. Na tabela seguinte conseguimos observar os valores de *accuracy* e AUC de cada modelo:

	Accuracy	AUC
Random Forest	0.6821	0.56
KNN	0.6762	0.53
Naïve Bayes	0.7061	0.54

Tabela 5 - Accuracy e AUC dos modelos com Stratified Sampling

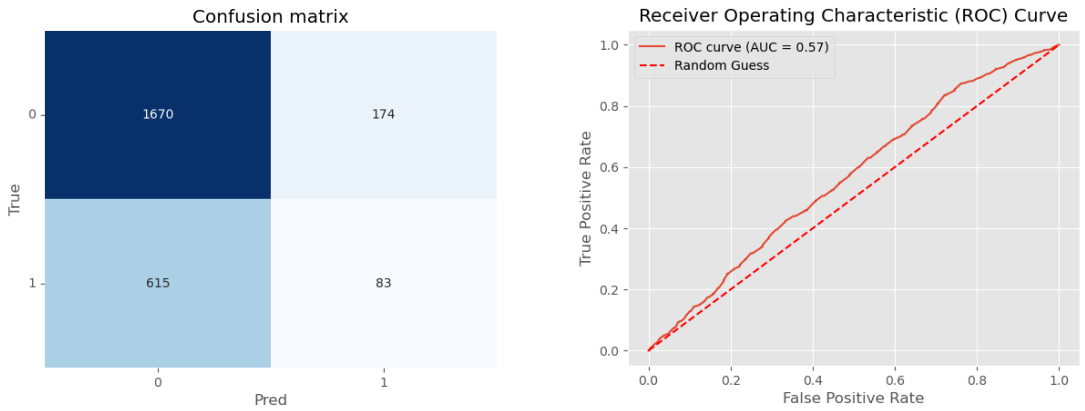

Figura 20 - Matriz de confusão e curva ROC do Random Forest

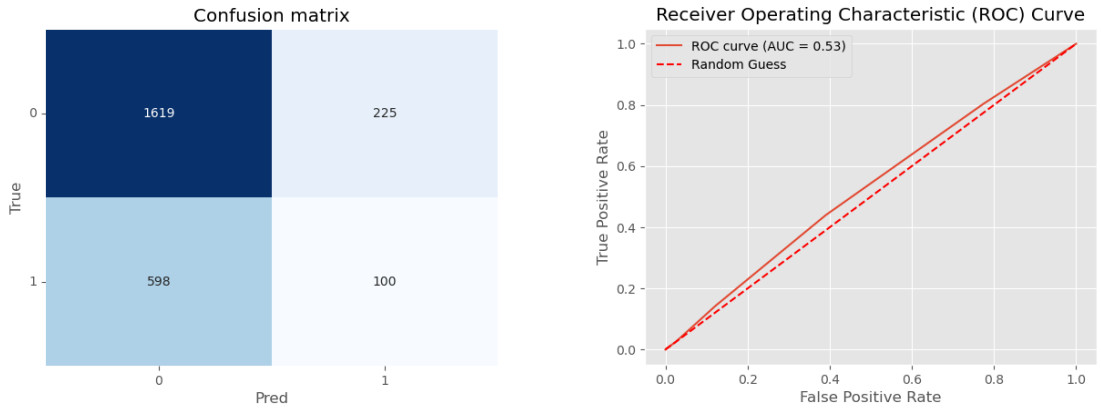

Figura 21 - Matriz de confusão e curva ROC do KNN

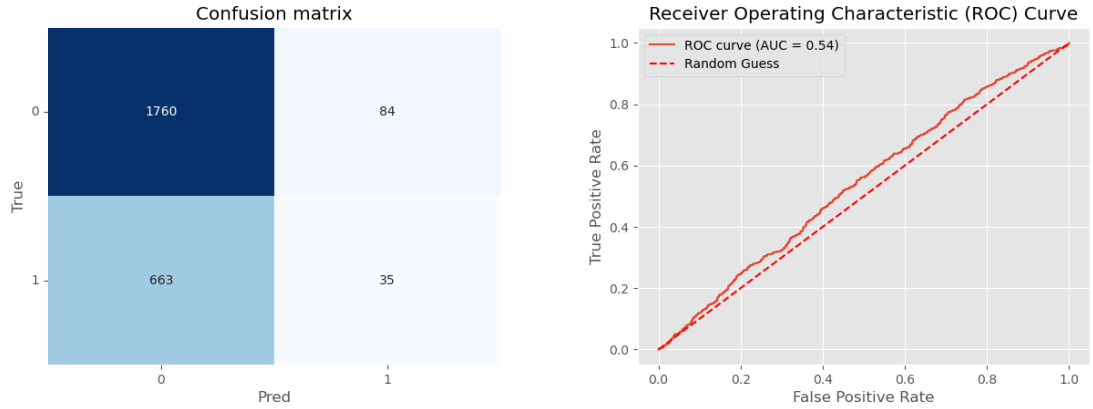

Figura 22 - Matriz de confusão e curva ROC do Naïve Bayes

4.1.4 Comparação de resultados e escolha do melhor modelo

A separação dos dados em conjuntos de treino e teste foi feita utilizando uma proporção de 80% para o conjunto de treino e 20% para o conjunto de teste. Essa divisão é normalmente utilizada para garantir que o modelo é testado numa amostra independente e representativa, tendo capacidade de generalização para novos dados. Foi testado o método de k-folds para dividir a base de dados, mas os resultados foram melhores com uma separação de 80/20, uma vez que o modelo, com o k-folds, só previa jogos com 2 sets, mesmo após aplicar o método de Stratified Sampling.

É possível observar os 3 resultados para cada uma das 3 estratégias anteriormente apresentadas na tabela seguinte:

		Accuracy	AUC
Oversampling	**Random Forest**	0.6310	0.57
	KNN	0.6884	0.54
	Naïve Bayes	0.7144	0.57
Undersampling	**Random Forest**	0.5496	0.58
	KNN	0.6884	0.54
	Naïve Bayes	0.7144	0.57
Stratified Sampling	**Random Forest**	0.6821	0.56
	KNN	0.6762	0.53
	Naïve Bayes	0.7061	0.54

Tabela 6 - Accuracy e AUC dos modelos com as 3 estratégias aplicadas

Comparando agora os resultados, e tendo em conta os pontos positivos e negativos de cada uma das técnicas, a técnica que foi escolhida foi a técnica de Stratified Sampling aplicada aos 3 modelos. Foi escolhida esta opção uma vez que combina bons valores de *accuracy* juntamente com bons valores de AUC, considerando, claro, todos os parâmetros positivos e negativos que cada técnica implica.

Dentro do método de Stratified Sampling, o modelo que foi escolhido foi o modelo Random Forest, uma vez que foi o modelo que pareceu mais equilibrado e mais robusto, que contém um valor de AUC superior aos restantes, mas que também tem um bom valor de *accuracy*.

4.2 Feature Selection

Os modelos anteriores foram executados com o conjunto de variáveis já limpas e bastante complexas, mas muitas delas podem não ser ideais para usar no modelo. Agora com o modelo ideal já selecionado, foram observadas as correlações das variáveis com a target e as importâncias de cada variável para o nosso modelo Random Forest.

4.2.1 Correlações

Devido à natureza das variáveis, foram usados dois métodos para calcular as correlações: o Rho de Pearson para as variáveis quantitativas e o V de Crammer para as variáveis qualitativas.

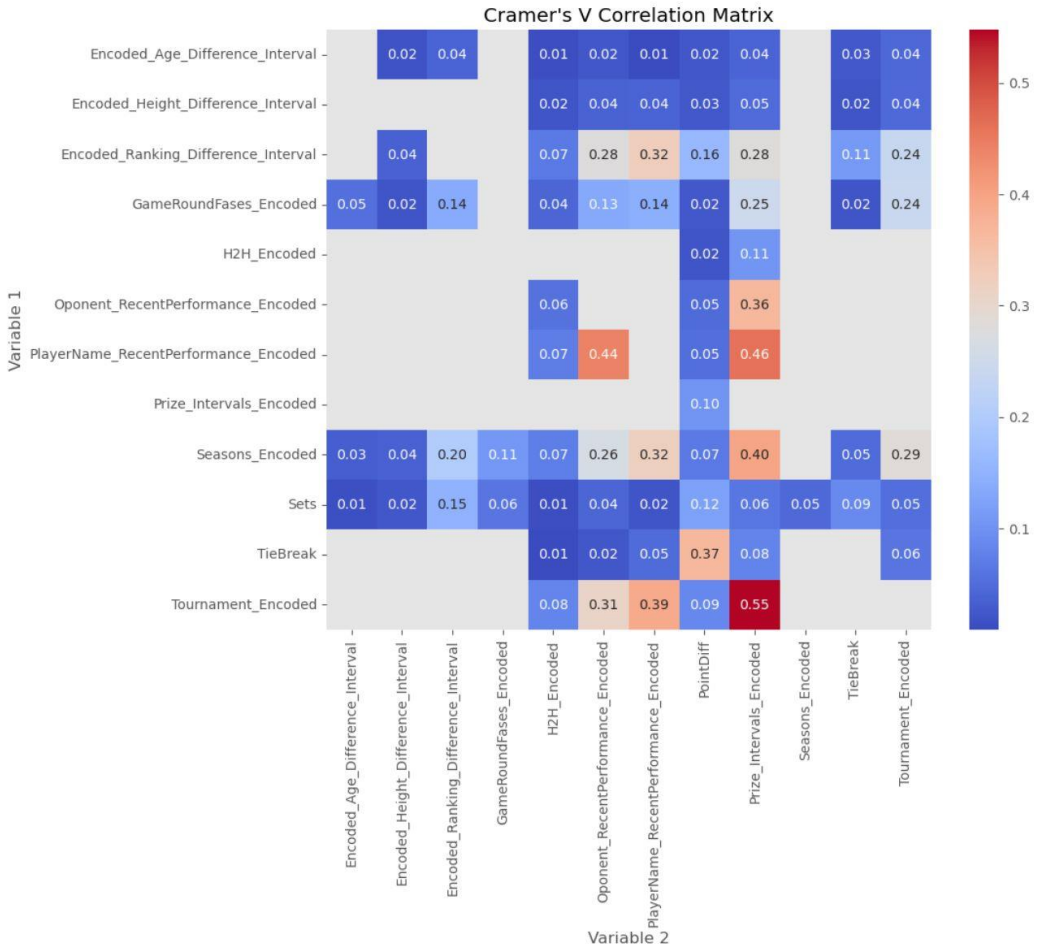

Figura 23 - Matriz das correlações de V de Crammer

Como se pode observar, existem correlações muito fracas com a variável 'Sets' sendo que maior parte delas não excede os 0.1 com exceção de duas variáveis: '**Encoded_Ranking_Difference_Interval**' e '**PointDiff**' com 0.15 e 0.12 respetivamente. Estão também presentes correlações moderadas entre as próprias variáveis como por exemplo,

entre os torneios e o prémio monetário, 0.55, ou até a performance mais recente do jogador com o prémio, 0.46, o que não surpreende, pois quanto maior o torneio mais provável é a quantia do prémio aumentar e quanto melhor a performance de um jogador, mais provável é este avançar no torneio e de ganhar o dinheiro.

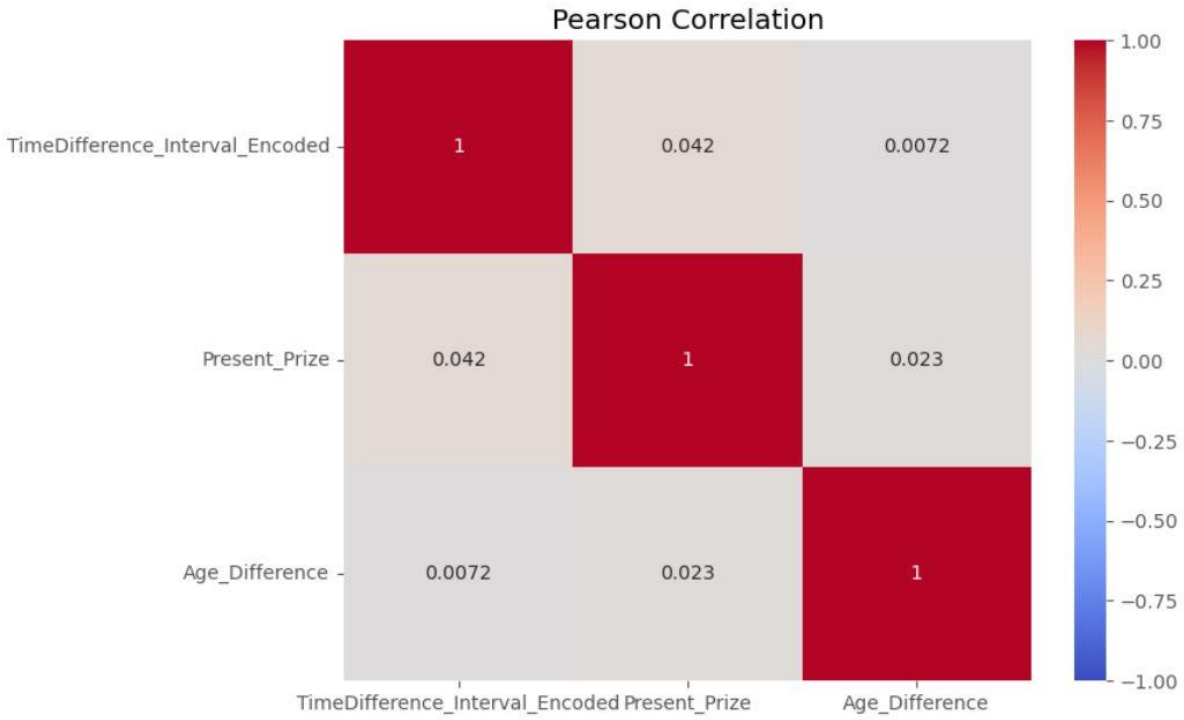

Figura 24 - Matriz das correlações de Pearson

Aqui podemos visualizar as correlações entre as variáveis numéricas e como anteriormente, as correlações são mesmo muito fracas sendo que a '**Age_Difference**' e '**TimeDifference_Interval_Encoded**' nem chega a 0.01, sendo que a melhor é entre '**Present_Prize**' e '**TimeDifference_Interval_Encoded**' com 0.04.

Para observar a importância e relevância de cada variável para o nosso modelo, foram calculadas as importâncias como é possível visualizar de seguida.

Índice da variável	Variável	Importância
13	PointDiff	0.108941
3	Encoded_Ranking_Difference_Interval	0.096790
12	Prize_Intervals_Encoded	0.086076
8	Oponent_RecentPerformance_Encoded	0.066441
10	TimeDifference_Interval_Encoded	0.061902
5	TieBreak	0.061882
9	H2H_Encoded	0.061860
2	Encoded_Age_Difference_Interval	0.060752
6	Tournament_Encoded	0.059704
1	GameRoundFases_Encoded	0.059150
11	Present_Prize	0.056179
4	Encoded_Height_Difference_Interval	0.055886
0	Seasons_Encoded	0.055299
7	PlayerName_RecentPerformance_Encoded	0.055240
14	Age_Difference	0.053898

Tabela 7 - Valores da importância das variáveis para o Random Forest (por ordem decrescente)

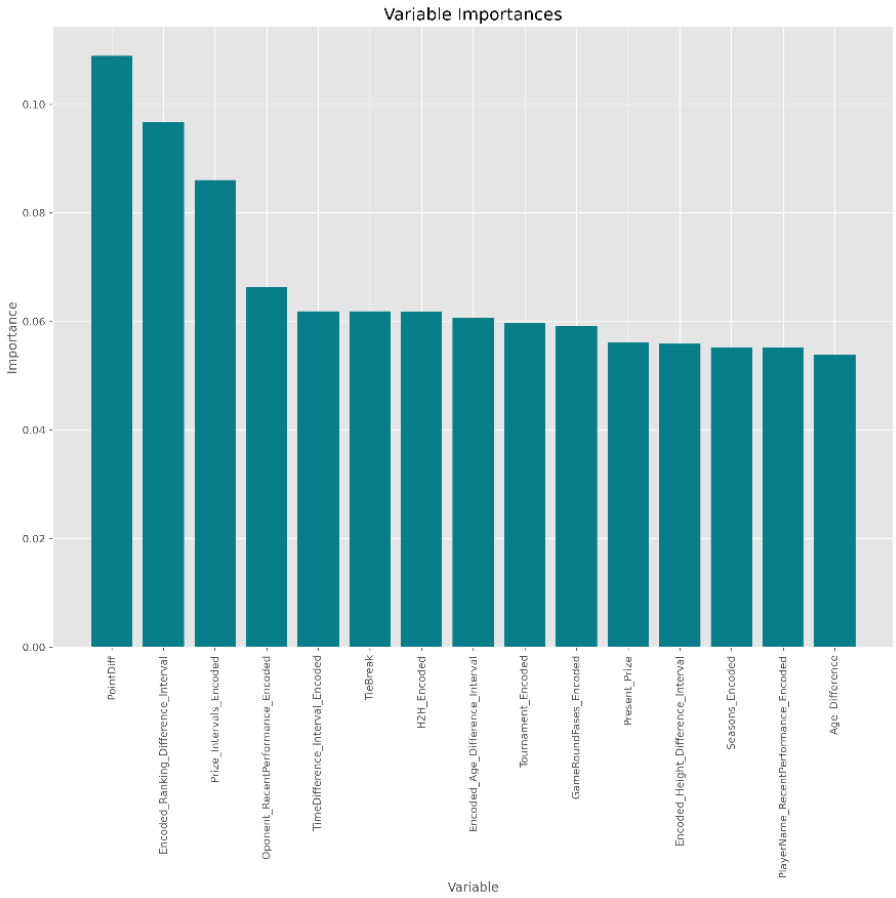

Figura 25 - Importância das variáveis para o modelo Random Forest (por ordem decrescente)

Estes valores de importância foram calculados usando uma função da library Scikit Learn chamada *'feature_importances'*. Visualizando a tabela e a figura anterior, foi tomada a decisão de manter as 7 variáveis com o maior valor de importância, ou seja, as 7 variáveis que mais impactam o nosso modelo Random Forest. Estas são as variáveis que foram selecionadas (para além da variável alvo *'Sets'*):

- 'PointDiff'

- 'Encoded_Ranking_Difference_Interval'

- 'Prize_Intervals_Encoded'

- 'Oponent_RecentPerformance_Encoded'

- 'TimeDifference_Interval_Encoded'

- 'TieBreak'

- 'H2H_Encoded'

4.3 Final Modeling

Os modelos finais foram então modelos Random Forest onde foi aplicada a técnica de Stratified Sampling e a divisão entre treino e teste foi de 80/20, respetivamente. Observando as variáveis selecionadas, é possível observar que estão presentes as variáveis 'PointDiff' e 'TieBreak'. Como é possível concluir no capítulo de Data Preparation, ambas as variáveis foram criadas a partir a variável Score, observando o resultado do primeiro set. Isto significa que, para ser possível prever o número de sets num determinado jogo e utilizar estas variáveis, é necessário que o jogo já esteja a decorrer e que o primeiro set já esteja concluído.

Devido a esta limitação do modelo, foram criados 2 modelos diferentes: um deles para prever o número de sets antes do jogo começar (todas as 7 variáveis anteriormente descritas, excluindo as variáveis 'PointDiff' e 'TieBreak'), e outro para prever o número de sets depois do primeiro set estar concluído (este já com as 7 variáveis).

4.3.1 Modelo Pré-Jogo

Este modelo contém então apenas 5 variáveis preditoras:

- 'Encoded_Ranking_Difference_Interval'
- 'Prize_Intervals_Encoded'
- 'Oponent_RecentPerformance_Encoded'
- 'TimeDifference_Interval_Encoded'
- 'H2H_Encoded'

Os resultados deste modelo de previsão Random Forest são (0 – 2 sets; 1- 3 sets):

Accuracy (overall correct predictions)	0.72
Auc	0.57
Recall (all 1s predicted right):	0.02
Precision (confidence when predicting a 1)	0.4

Tabela 8 - Métricas para avaliação do modelo Pré-Jogo

	precision	recall	f1-score	support
0	0.73	0.99	0.84	1844
1	0.40	0.02	0.05	698

Tabela 9 - Métricas para avaliação do modelo Pré-Jogo para cada classe

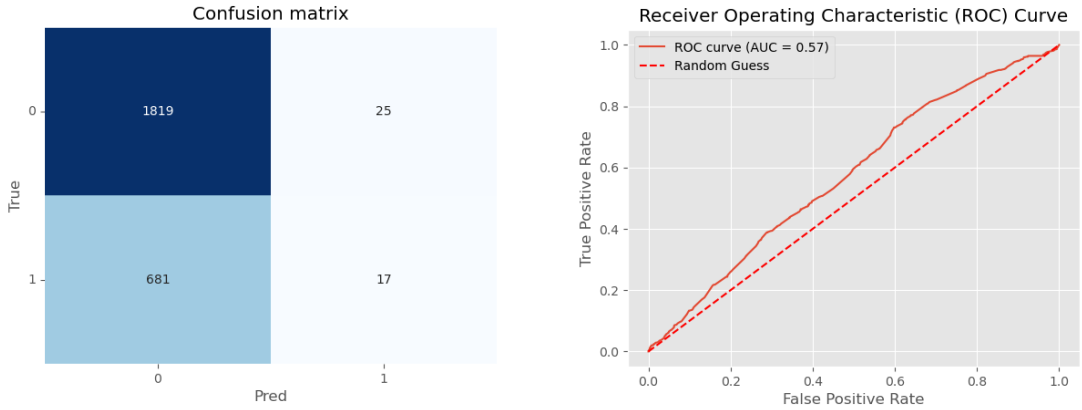

Figura 26 - Matriz de confusão e curva ROC do modelo Pré-Jogo

4.3.1 Modelo Pós-Primeiro Set

Este modelo contém as 7 variáveis preditoras anteriormente propostas:

- 'PointDiff'

- 'Encoded_Ranking_Difference_Interval'

- 'Prize_Intervals_Encoded'

- 'Oponent_RecentPerformance_Encoded'

- 'TimeDifference_Interval_Encoded'

- 'TieBreak'

- 'H2H_Encoded'

Os resultados deste modelo de previsão Random Forest são (0 – 2 sets; 1- 3 sets):

Accuracy (overall correct predictions)	0.71
AUC	0.58
Recall (all 1s predicted right):	0.06
Precision (confidence when predicting a 1)	0.33

Tabela 10 - Métricas para avaliação do modelo Pós-Primeiro Set

	precision	recall	f1-score	support
0	0.73	0.96	0.83	1844
1	0.33	0.06	0.10	698

Tabela 11 - Métricas para avaliação do modelo Pós-Primeiro Set para cada classe

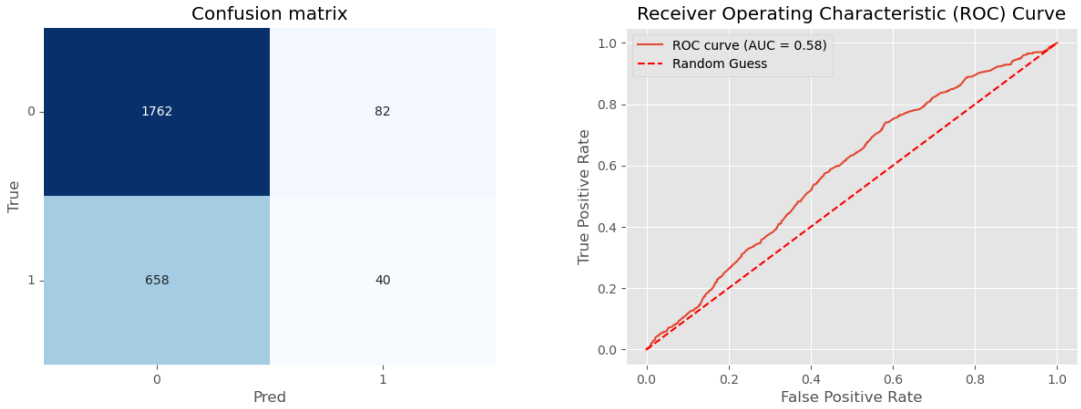

Figura 27 - Matriz de confusão e curva ROC do modelo Pós-Primeiro Set

5. Evaluation

Observando e analisando os resultados, o modelo 1 teve 72% de previsões corretas no geral. Teve um valor da AUC de 0.57, que indica a capacidade de o modelo classificar corretamente as instâncias positivas e negativas. Quanto maior o valor, melhor o desempenho. Teve uma taxa de recall muito baixa, indicando que teve dificuldade em identificar corretamente os jogos com 3 sets (valor 1). Teve uma precisão de 0.4, que corresponde à proporção de jogos com 3 sets que previu corretamente em relação ao total de instâncias previstas como positivas. Neste modelo, a precisão também é relativamente baixa.

O modelo 2 teve uma taxa de previsões corretas de 71% no geral. O valor da AUC foi de 0.58, o que mostra um desempenho ligeiramente melhor quando comparado com o modelo 1. Com um recall de 0.06, este modelo conseguiu identificar ligeiramente melhor os jogos com 3 sets em comparação com o modelo 1, mas ainda apresenta uma taxa de *recall* baixa. A precisão foi de 0.33, apresentando uma precisão baixa, o que indica uma taxa relativamente alta de falsos positivos.

Ambos os modelos têm dificuldades em lidar com a classe positiva (1) corretamente, que corresponde aos jogos com 3 sets, tendo baixos valores de *recall* e precisão. Considerando ambos os resultados, o modelo 1 apresenta resultados um pouco melhores em termos de *recall* e precisão em comparação com o modelo 2, mas ainda há espaço para melhorias significativas em ambos os casos.

Fazendo agora uma análise mais detalhada do modelo 1,

Para a classe 0 (2 sets):

- Precision: 0.73; o modelo teve uma taxa relativamente alta de previsões corretas para jogos disputados em 2 sets.

- Recall: 0.99; indica uma alta taxa de identificação correta dos jogos disputados em 2 sets em relação ao total de jogos dessa classe.

- F1-score: 0.84; é uma medida que combina Precision e Recall, fornecendo uma medida geral do desempenho da classe 0. Isto significa, então, que o modelo está equilibrando relativamente à Precision e ao Recall para a classe 0. Isto significa que o modelo está a identificar corretamente os jogos disputados em 2 sets, considerando tanto as previsões corretas (Precision) como a taxa de identificação correta em relação ao total de jogos dessa classe (Recall).

Para a classe 1 (3 sets):

- Precision: 0.40; o modelo teve uma taxa relativamente baixa de previsões corretas para jogos disputados em 3 sets.

- Recall: 0.02; indica uma taxa muito baixa de identificação correta dos jogos disputados em 3 sets em relação ao total de jogos dessa classe.

- F1-score: 0.05, indica um baixo desempenho geral na previsão da classe 1.

Analisando o modelo 2,

Para a classe 0 (2 sets):

- Precision: 0.73; o modelo apresenta uma precisão semelhante ao modelo 1 para a classe 0.

- Recall: 0.96; mostra uma taxa relativamente alta de identificação correta dos jogos disputados em 2 sets em relação ao total de jogos dessa classe.

- F1-score: 0.83; indica um bom desempenho geral na previsão da classe 0, ou seja, jogos de 2 sets.

Para a classe 1 (3 sets):

- Precision: 0.33; o modelo tem uma taxa relativamente baixa de previsões corretas para jogos disputados em 3 sets.

- Recall: 0.06; indica uma taxa baixa de identificação correta dos jogos disputados em 3 sets em relação ao total de jogos dessa classe.

- F1-score: 0.10; indica um desempenho geral baixo na previsão da classe 1, mas ainda assim superior ao do modelo anterior.

Concluindo, ambos os modelos têm dificuldades em identificar corretamente os jogos disputados em 3 sets (classe 1), apresentando valores mais baixos de recall, precisão e F1-score para essa classe. O Modelo 1 obteve uma taxa de recall mais baixa do que o Modelo 2, revelando assim a dificuldade que tem em identificar jogos disputados em 3 sets corretamente, comparado com o Modelo 2. No entanto, o Modelo 1 apresenta uma precisão ligeiramente melhor para essa classe. Tendo em conta estes fatores e as vantagens, desvantagens, limitações que ambos os modelos apresentam, é difícil escolher qual o melhor modelo para a previsão do número de sets.

Conclusão

O objetivo proposto pelos docentes foi prever o número de sets de um jogo de ténis. Foi utilizada a metodologia CRISP-DM na totalidade do trabalho, o que nos permitiu uma melhor organização e uma melhor gestão do tempo e do trabalho em geral. Foram feitos vários testes para fundamentar todas as escolhas tomadas ao longo do trabalho, não só na limpeza das variáveis, como na criação e seleção de outras novas variáveis, e também na modelagem.

Mesmo recorrendo a bases de dados externas e criando variáveis, os modelos desenvolvidos não têm as capacidades de previsão ideais, e uma vez que no contexto do problema onde os modelos seriam usados para apostas, os modelos finais podem não ser suficientes para casas de apostas. Ainda assim, o resultado final é satisfatório e pode ser bastante útil para diversos casos.

Com a realização deste projeto foi possível compreender que é muito difícil fazer boas previsões de problemas reais. No mundo real, existem imensos fatores que podem influenciar a decisão final. No desporto, e neste caso no ténis, isto não é exceção. Não só fatores como o clima, espaço, altura, peso, (entre outros), podem influenciar o resultado, como também outros fatores humanos que os dados não conseguem mostrar, como a concentração e pressão que um jogador possa estar a sentir, o cansaço, o estado de espírito, entre outros.

Este projeto trouxe muitos desafios não só na parte mais prática e teórica, como a limpeza da base de dados, mas também na parte mais crítica e criativa, onde foi necessário conhecer bem a área e compreender que novas variáveis eram possíveis criar e adaptar para aumentar a precisão do modelo. Por vezes houve a necessidade de refazer passos e voltar atrás em algumas decisões tomadas pois à medida que fomos trabalhando com a base de dados foram surgindo novas ideias e novas limitações.

Apesar de todas estas limitações e dificuldades sentidas, consideramos que os modelos alcançados aliados ao conhecimento de ténis e espírito crítico de cada pessoa, formam uma ferramenta útil para os entusiastas das apostas desportivas, podendo tomar decisões mais firmes e estratégicas.

Referências Bibliográficas

- History. ATP Tour Tennis. (n.d.). ATP Tour. https://www.atptour.com/en/corporate/history;

- Official Site of Men's Professional Tennis. ATP Tour Tennis. (2019). ATP Tour. https://www.atptour.com/en;

- Datopian. (n.d.). ATP World Tour tennis data. DataHub. https://datahub.io/sports-data/atp-world-tour-tennis-data#data-cli;

- Definitions USTA Mississippi. (n.d.). Mstennis. https://mstennis.com/content/definitions;

- Sinkovic, F., Novak, D., Foretic, N., & Zemková, E. (2023). The Effects of Biological Age on Speed-Explosive Properties in Young Tennis Players. Journal of Functional Morphology and Kinesiology, 8(2), 48. https://www.mdpi.com/2411-5142/8/2/48;

- Crim, J. (2014). *Types of Tennis Shots | An Overview Of The Different Strokes In Tennis*. Tenniscompanion.org. https://tenniscompanion.org/types-of-tennis-shots/;