Universidade Federal de Uberlândia
Faculdade de Matemática

Bacharelado em Estatística

ESTUDO DO JOGO DE PÔQUER UTILIZANDO MÉTODOS ESTATÍSTICOS E APRENDIZADO DE MÁQUINA

Lucas de Oliveira Diniz

Uberlândia-MG
2022

Lucas de Oliveira Diniz

ESTUDO DO JOGO DE PÔQUER UTILIZANDO MÉTODOS ESTATÍSTICOS E APRENDIZADO DE MÁQUINA

Trabalho de conclusão de curso apresentado à Coordenação do Curso de Bacharelado em Estatística como requisito parcial para obtenção do grau de Bacharel em Estatística.

Orientador: Prof. Dr. Pedro Franklin Cardoso Silva

Uberlândia-MG

2022

Universidade Federal de Uberlândia
Faculdade de Matemática

Coordenação do Curso de Bacharelado em Estatística

A banca examinadora, conforme abaixo assinado, certifica a adequação deste trabalho de conclusão de curso para obtenção do grau de Bacharel em Estatística.

Uberlândia, ____ de _____ de 20____

BANCA EXAMINADORA

_____

Prof. Dr. Pedro Franklin Cardoso Silva

_____

Prof. Dr. Leandro Alves Pereira

_____

Prof. Dr. José Waldemar da Silva

Uberlândia-MG

2022

Resumo

O presente trabalho foi desenvolvido com a finalidade de modelar ações de jogadores no pôquer e classificá-los como vencedores ou perdedores utilizando métodos estatísticos e de aprendizado de máquina. Para aplicar essas técnicas, utilizamos uma base de dados real referente a aproximadamente 22 milhões de partidas disputadas em cassinos online. Recorremos ao software Holdem Manager para pré-processamento dos dados e a aplicação em algoritmos foi realizada no R, considerando 13 variáveis explicativas que representam ações dos jogadores. Criamos um conjunto em que a proporção de jogadores vencedores e perdedores seja aproximadamente a mesma. Em seguida a base foi divida. 80% das observações foram para um conjunto de treinamento e 20% das observaçoes foram para um conjunto de teste. O modelo final de classificação usando Floresta Aleatória obteve uma acurácia de 0,75. Identificamos que as ações estratégias são importantes para a vitória no pôquer, isto é, os jogadores que tendem a ser ganhadores se comportam de um modo diferente dos perdedores no que se diz respeito a frequência de realizar movimentos no jogo.

Palavras-chave: Pôquer, Aprendizado de Máquina, Árvore de Decisão, Floresta Aleatória.

Abstract

The present work was developed with the purpose of modeling actions of players in the game of poker and classify them as winners or losers using statistical and machine learning methods. To apply these techniques, we used a real database that contains around 22 million games played at online casinos. We appeal to Holdem Manager software for data pre-processing and application in algorithms was performed in R, considering 13 explanatory variables that represent players actions. We create a data set in which the ratio of winning and losing players is approximate the same, then the base was divided into 80% for training and 20% for testing and validate the model. The final classification model using Random Forest obtained an accuracy ratio of 0.75. We identified that strategic actions are important for winning in poker, that is, players who tend to be winners behave differently from losers with regard to the frequency of making moves in the game.

Keywords: Poker, Machine Learning, Decision Tree, Random Forest.

Sumário

Lista de Figuras

Lista de Tabelas

1. Introdução

Os jogos de estratégia apresentam elevado potencial para pesquisas científicas envolvendo estatística computacional pois apresentam características como um conjunto de regras pré-estabelecidas e objetivos específicos. O bom desempenho nesses jogos é um desafio que pode ser auxiliado pela aplicação de conceitos teóricos em algoritmos e a performance obtida pode ser avaliada por resultados quantitativos [2].

O pôquer é um jogo competitivo amplamente praticado ao redor do mundo, seja na modalidade presencial ou em ambientes virtuais. Através de um computador, smartphone ou outros dispositivos com capacidade de se conectar à internet, um usuário pode disputar partidas on-line com adversários de vários países.

Em 2009, uma das plataformas que disponibiliza o pôquer online contou com mais de 300 mil participantes jogando simultaneamente [16]. Além da opção de praticar com moedas fictícias, os jogadores também podem, diariamente, realizar apostas envolvendo dinheiro real. Um único campeonato disputado pela internet já retornou o prêmio de 27.559.500,00 dólares americanos para o vencedor [17].

Devido ao pôquer envolver apostas e existir uma crença do fator sorte, ainda hoje o jogo é equivocadamente associado por alguns como um jogo de azar. O reconhecimento oficial do pôquer como esporte mental ocorreu em 2010, na assembléia realizada pela IMSA (International Mind Sports Association) [13], atribuindo condição similar ao xadrez, bridge e ao go. Embora existam praticantes recreativos, há outros que tratam o pôquer como profissão e conseguem se destacar obtendo resultados consistentes através do aprofundamento na teoria e desenvolvimento de habilidades práticas.

Parte da teoria do jogo inclusive é ministrada em disciplinas pela Universidade Estadual de Campinas (UNICAMP) [21] e pelo Instituto de Tecnologia de Massachusetts (MIT) [4]. Outras universidades possuem um grupo de pesquisa específico voltado ao pôquer, como o University of Alberta Computer Poker Research Group [1] e o Departamento de Ciência da Computação da Carnegie Melon University, onde foi desenvolvido o "Libratus"[10], o primeiro programa de computador que venceu quatro grandes especialistas humanos do esporte em uma disputa realizada em 2017 (humans vs I.A match) [23]. Para a construção do programa foi utilizado o método de Counterfactual Regret Minimization (CFR) [22]. Os algoritmos iterativos desse método contém estratégias de equilíbrio de Nash e demandam elevados recursos computacionais [8].

Como veremos adiante na fundamentação teórica desse projeto, o pôquer é um jogo que

envolve cartas e fichas e não sabemos as cartas do(s) nosso(s) adversário(s), ou seja, lidamos com probabilidade de eventos aleatórios e com processos de tomada de decisão. Fenômenos que apresentam essas características são temas de grande interesse para as áreas de estatística computacional relacionadas a inteligência artificial e mineração de dados.

Métodos estatísticos (Multivariada), probabilísticos (Monte Carlo) e de aprendizado de máquina (Árvore de Decisão, Floresta Aleatória, Análise de Cluster) são ferramentas importantes que podem ser usadas para modelar e classificar os dados a fim de reconhecer padrões de associações e estimar probabilidades. O estúdo desses métodos usando o exemplo do pôquer pode proporcionar um aprofundamento nas teorias de probabilidade e uma otimização no processo de decisão não só para os interessados em melhorar sua estratégia no jogo mas também em outras circunstâncias de atuação social e profissional [20].

Na primeira parte desse trabalho apresentaremos as regras básicas do jogo, em seguida desenvolveremos as teorias da probabilidade, dos métodos estatísticos e dos métodos de aprendizado de máquina que utilizaremos para aplicação dessas teorias em algoritmos. Para aplicarmos essas técnicas, utilizaremos uma base de dados real referente a aproximadamente 22 milhões de partidas disputadas em cassinos online [3]. Após realizarmos os pré-processamentos necessários, selecionaremos os atributos relevantes do conjunto de dados, faremos o reconhecimento de padrões e modelaremos ações de oponentes para classificar um jogador como vencedor ou perdedor através de algoritmos de Árvore de Decisão e Floresta Aleatória.

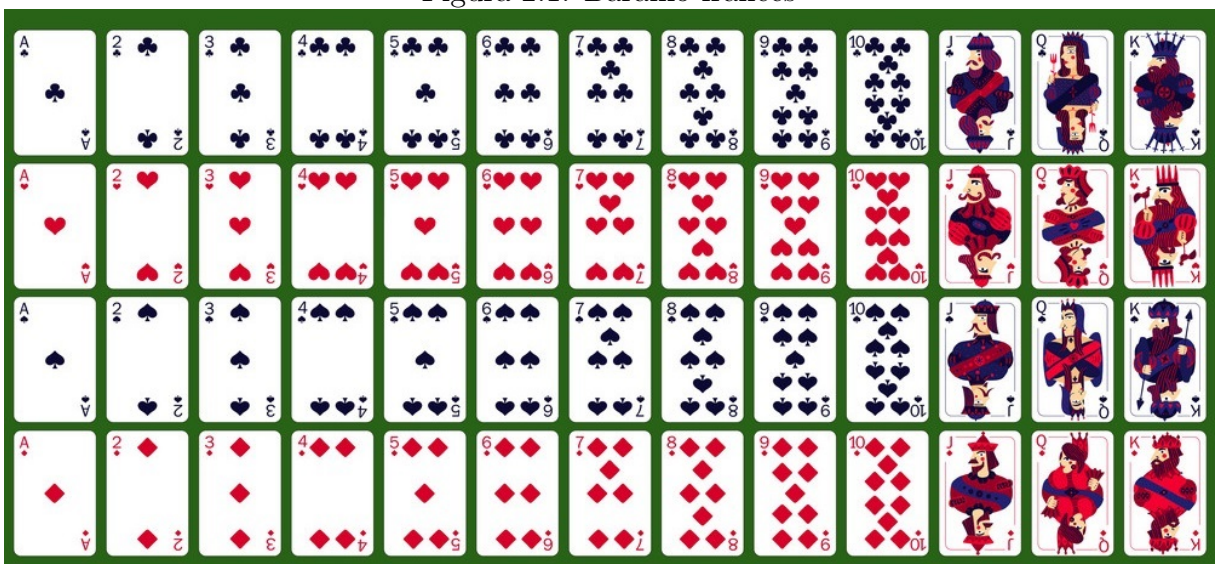
2. Fundamentação Teórica

Neste capítulo apresentaremos o embasamento teórico desse trabalho, desde as regras gerais do pôquer até as técnicas de mineração de dados e aprendizado de máquina a serem utilizadas no seu desenvolvimento.

2.1 Regras básicas do pôquer

Embora existam diversas modalidades de pôquer, quase todas envolvem apostas e cartas. As apostas usualmente são feitas utilizando fichas, dinheiro fictício ou dinheiro real. O baralho principal é chamado baralho francês e contém 52 cartas, sendo 13 de cada um dos quatro naipes diferentes (paus, copas, espadas e ouros), ele pode ser visualizado na Figura 2.1.

Figura 2.1: Baralho francês



Fonte: VectorStock.com/20600919

Os jogadores competem entre si em uma mesa formada geralmente por 2 a 10 integrantes. Durante cada partida ocorrem rodadas de apostas. O participante pode vencer uma partida e ganhar o montante das apostas realizadas (pote) revelando a melhor combinação de cartas ou através da desistência dos oponentes.

A modalidade que focaremos nesse trabalho é chamada No Limit Texas Hold'em com mesas de 6 jogadores. Ela foi escolhida por ser a mais popular.

2.1.1 NO-LIMIT TEXAS HOLD'EM

Inicialmente, cada mesa de jogo possui um valor de aposta mínima (big blind) e cada participante tem um montante de fichas denominado stack. A característica da vertente No-Limit é que os jogadores podem realizar apostas de qualquer valor entre o big blind e o stack.

Na variação Texas Hold'em, ou apenas Hold'em, 2 cartas são distribuídas para cada indivíduo e apenas ele pode vê-las. Em seguida inicia-se as rodadas de apostas e 5 cartas comunitárias podem ser expostas. Na Figura 2.2 podemos observar alguns elementos de uma mesa de Texas Hold'em online.

Figura 2.2: Início de uma partida de No Limit Texas Hold'em

Fonte: Mesa observada no cassino online ggpoker em 17/03/2022

2.1.2 Sistema de apostas

Cada jogador tem sua vez de agir e a ordem se dá no sentido horário, sendo o ponto de partida dependente de quem foi o embaralhador (dealer). As opções de ação disponíveis são:

- **Bet:** Realizar uma aposta, caso ninguém tenha feito. Como mencionado anteriormente, na modalidade de No-Limit Hold'em, o valor mínimo de uma aposta deve ser do mesmo tamanho que o big-blind e o valor máximo pode ser tão alto quanto a quantidade de fichas que o jogador tem na mesa. Apostar todas as fichas é denominado All-in.
- **Raise:** Aumentar o valor da aposta que alguém fez anteriormente.
- **Call:** Pagar a aposta que alguem fez anteriormente. Caso nenhum oponente pague a aposta (ou aumente), o último jogador restante ganha o pote.
- **Fold:** Desistir é o ato de descartar as cartas da mão e perder a rodada. Quando um jogador desiste ele abandona a oportunidade de disputar o pote.
- **Check:** Passar a vez sem desistir da mão. Check é basicamente o mesmo que pagar uma quantia de 0. Caso uma aposta tenha sido feita, os jogadores que enfrentaram essa aposta não podem mais passar a vez.

2.1.3 RODADA INICIAL E CARTAS COMUNITÁRIAS

Uma única partida (mão), pode ser dividida em quatro etapas centrais, denominadas streets.

- **PRÉ-FLOP:** Inicialmente, os dois jogadores a esquerda do dealer fazem apostas fixas obrigatórias (small blind e big blind). Em seguida todos os integrantes da mesa recebem suas duas cartas privadas e podem escolher entre as ações mencionadas anteriormente, exceto Bet, pois os blinds ja representam uma primeira aposta.

Figura 2.3: Exemplo de sequência de ações no pré-flop

Fonte: Elaborada pelo autor utilizando o software holdem manager [6]

No exemplo da Figura 2.3 percebemos através do indicador no canto inferior esquerdo a sequência de ações dessa rodada. O jogador P2 aumentou (raise) a aposta obrigatória dos blinds e todos os demais desistiram (fold), exceto o Hero, que igualou (call) o valor aumentado. Como não restaram mais jogadores para agir, as primeiras cartas comunitárias são expostas para os envolvidos.

- **FLOP:** Três cartas comunitárias são abertas na mesa para os participantes remanescentes, caso houverem. Inicia-se outra rodada de apostas.

Figura 2.4: Exemplo de sequência de ações no flop

Fonte: Elaborada pelo autor utilizando o software holdem manager [6]

A sequência de ações nessa rodada também foi registrada e está sinalizada pela seta na Figura 2.4. Como a ordem de ação ocorre no sentido horário a partir da posição do dealer, o primeiro a agir foi o Hero, que optou por passar a vez, levando a ação para o P2, que é o próximo e único jogador ainda presente na mão. P2 fez uma aposta, a ação voltou para o Hero, que optou por aumentar. Por fim o P2 igualou o valor aumentado e os dois jogadores seguiram para a próxima rodada.

- **TURN:** Uma quarta carta comunitária é virada na mesa quando a rodada de apostas no flop é encerrada e mais de um jogador continua na mão. Outra rodada de apostas se inicia.

Figura 2.5: Exemplo de sequência de ações no turn

Fonte: Elaborada pelo autor utilizando o software holdem manager [6]

No exemplo da Figura 2.5, após a abertura da penúltima carta comunitária, o Hero tomou a ação de apostar, o jogador P2 aumentou essa aposta e em seguida o Hero pagou o aumento. Como não restaram mais jogadores para agir e não houve desistência entre os envolvidos, a próxima carta comunitária é exposta e inicia-se outra sequência de ações.

- **RIVER:** A quinta e última carta é aberta na mesa e ocorre a última rodada de apostas entre os envolvidos que restaram. Após essa rodada, caso não haja desistência, os jogadores apresentam suas duas cartas e quem tiver o melhor jogo vence, ganhando todo o montante de apostas.

Figura 2.6: Exemplo de sequência de ações no river

Fonte: Elaborada pelo autor utilizando o software holdem manager [6]

Através da Figura 2.6, observamos que a primeira escolha foi do Hero, que optou por passar a vez. Em seguida, o P2 apostou todas as suas fichas (All-in), e ação voltou para o Hero, que escolheu pagar. Nessa última decisão, percebemos que o Hero não tem a opção de aumentar, apenas pagar ou desistir, pois o seu oponente já apostou todas suas fichas. Podemos observar também que o valor apostado pelo P2 é maior que o montante de fichas total restantes para o Hero, portanto o excedente retorna para o apostador. Em uma única ação da mesa, o máximo que um jogador pode apostar é o seu stack.

Como não existem mais ações disponíveis e não houve desistência, nesse momento os jogadores apresentam seus jogos e quem tiver a melhor combinação de cartas ganha as fichas que acumularam na mesa durante essa partida.

- **SHOWDOWN:** Showdown é o nome dado para o momento em que os jogadores envolvidos revelam suas cartas, como podemos observar na Figura 2.7. Pode ocorrer o showdown antes do river caso os participantes tenham apostado todas suas fichas. Nesse caso as 5 cartas comunitárias ainda são viradas (sem rodadas de apostas) e vence quem formar o melhor jogo no final. O ranking de força de jogo será apresentado na sessão 2.1.4.

Figura 2.7: Exemplo de showdown no river

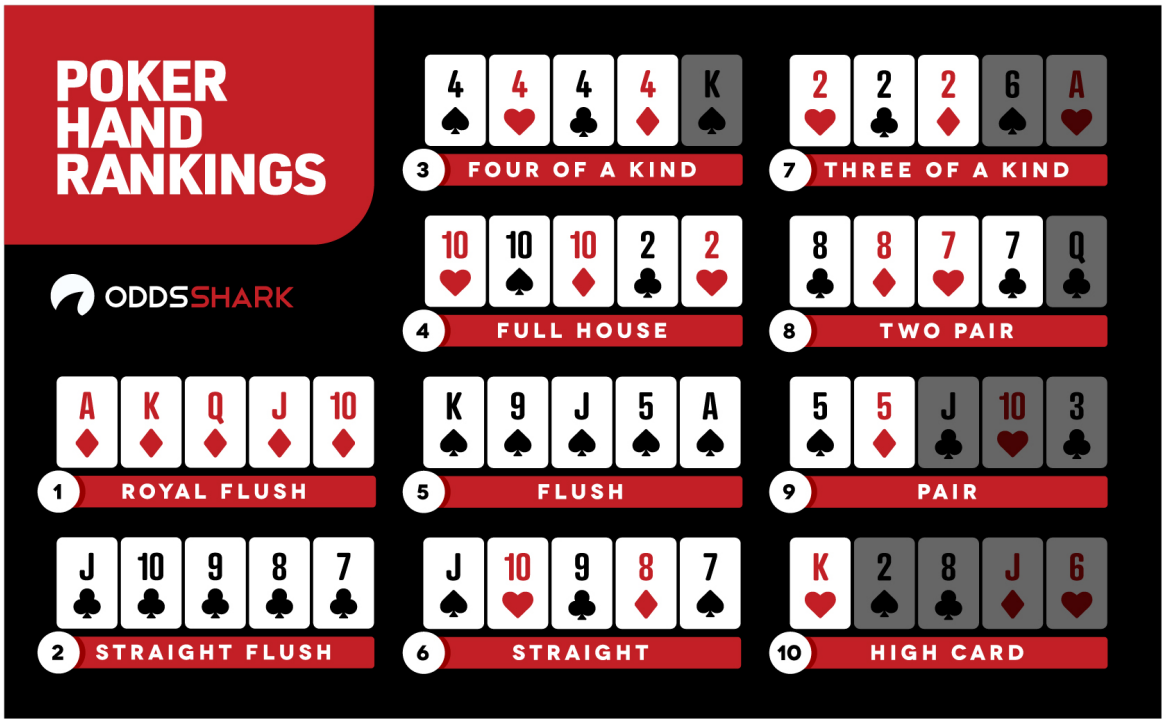
Fonte: Elaborada pelo autor utilizando o software holdem manager [6]

Um estudo chamado "Statistical Analysis of Texas Hold'em" realizado em 2009 por MP Hope considerando uma base de dados real de 103 milhões de partidas revelou que apenas 24.3% dessas mãos chegaram até o showdown. Esse resultado é um indício que a maioria dos jogos são determinados por algo além da força de mão, como a habilidade de blefar, forçando a desistência dos oponentes e ganhando as fichas do pote [11].

2.1.4 Ranking de mãos

Nas variações de pôquer que usam cartas comunitárias, a classe de jogo final do jogador é a melhor combinação possível de suas cartas individuais com as cartas comunitárias. No caso do Texas Hold'em, a força do jogo é medida pela melhor combinação de 5 cartas dentre as 7 disponíveis. Na Figura 2.9 a seguir temos as classificações possíveis, em ordem decrescente de força.

Figura 2.8: Ranking da força do jogo no Texas Hold'em



Fonte: Imagem extraída de oddsshark.com

- **Royal flush:** Sequência do 10 ao Ás do mesmo naipe.
- **Straight flush:** Cinco cartas consecutivas do mesmo naipe.
- **Four of a kind:** Quatro cartas de mesmo valor.
- **Full house:** Três cartas de mesmo valor (trinca) e duas outras de mesmo valor (par). Caso dois ou mais jogadores tenham full house, vence quem tiver a trinca mais alta. Se empatarem na trinca, vence o maior par.
- **Flush:** Cinco cartas do mesmo naipe. Se houver empate, vence quem tiver a carta mais alta do Flush. Se persistir, vence quem tiver a segunda, a terceira, quarta ou mesmo a quinta carta mais alta.
- **Straight:** Cinco cartas consecutivas de qualquer naipe.

- **Three of a kind:** Três cartas de mesmo valor. Caso mais de um jogador tenha trinca, vence a mais alta. Se a trinca for a mesma para mais de um jogador, vence quem tiver a primeira carta mais alta fora da trinca. Se o empate persistir, a decisão é pela segunda mais alta. Uma carta que não participa da classificação da mão, mas é utilizada para desempate entre mãos iguais é chamada de kicker.

- **Two pair:** Duas cartas de mesmo valor mais duas outras cartas de mesmo valor. Em caso de empate, vence o par mais alto. Persistindo, o segundo par mais alto. Se dois jogadores tiverem dois pares iguais, vence aquele que tiver o kicker mais alto.

- **Pair:** Duas cartas de valores iguais. Em caso de empate, kicker mais alto da terceira, da quarta ou quinta carta, sucessivamente.

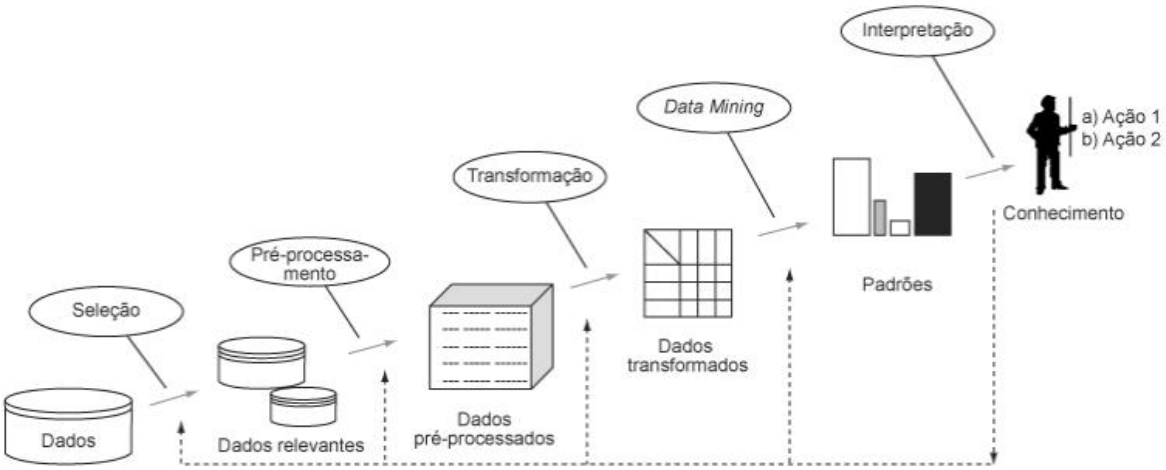
- **High card:** Se o showdown revelar que nenhum jogador tem uma combinação, vence quem tiver a carta mais alta. Em caso de empate, usa-se da segunda à quinta carta como critério para saber quem fica com o pote.

Cada combinação final é formada por um total de 5 cartas. Portanto, mesmo que a classe de jogo seja por exemplo two pair (dois pares), a quinta carta também será considerada e é chamada kicker. A ordem crescente de força das cartas é 2,3,4,5,6,7,8,9,10,J,Q,K,A, sendo que o A também pode compor sequências de A,2,3,4,5.

2.2 MINERAÇÃO DE DADOS

Mineração de dados, também conhecida pelo termo em inglês "datamining" é o processo de descobrir informações úteis em um grande repositório de dados. A exploração dos dados é feita de maneira automática ou semi-automática com o auxílio de técnicas estatísticas e computacionais [12]. Os algoritmos associados aos programas de computador fazem manipulações do bancos de dados buscando regularidades ou padrões. Padrões significativos, se encontrados, provavelmente se generalizarão para fazer previsões sobre dados futuros [7]. Na Figura 2.9 podemos observar um diagrama que representa o processo de mineração de dados.

Figura 2.9: Processo para descoberta de conhecimento nos dados



Fonte: Fayyad U. et al. Knowledge Discovery and Data Mining, KDD 96, 1996

As tarefas de mineração de dados podem ser divididas em dois tipos: tarefas descritivas e preditivas. As tarefas descritivas encontram padrões que descrevem os dados, enquanto as preditivas usam algumas variáveis para prever valores desconhecidos ou futuros de outras variáveis. Existem dois tipos de modelos descritivos: as regras de associação e os métodos de agrupamento.

TAREFAS DESCRITIVAS

- **Regras de associação:** São utilizadas para encontrar padrões de associação entre os dados. Geralmente são representados por regras de implicação. Na Figura 2.10 ilustramos um exemplo clássico na literatura de mineração de dados. Ao avaliar transações de compras feitas por clientes de supermercado, a regra de associação obtida indicou que clientes que compraram fraldas tenderam a comprar cerveja [12]. Uma notação apropriada para indicar essa regra é:

  Regra 1: Cerveja $\Longrightarrow$ Fralda.

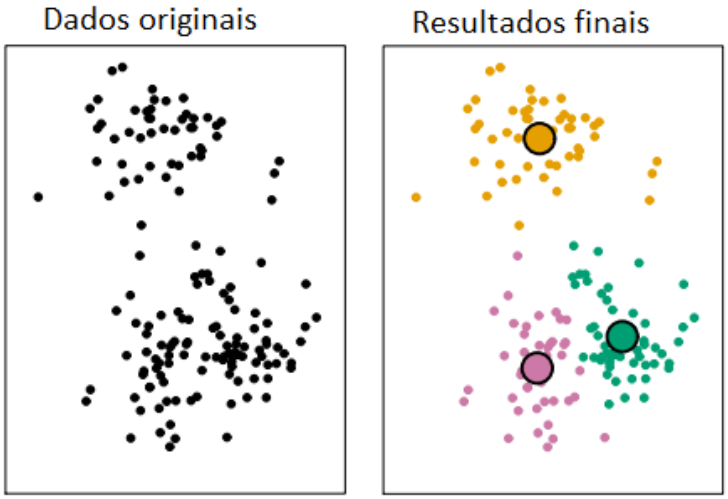
Figura 2.10: Regras de Associação em transações de supermercado

Transação	Itens
1	{Pão, Leite}
2	{Pão, Fraldas, Cerveja, Ovos}
3	{Leite, Fraldas, Cerveja, Refrigerante}
4	{Pão, Leite, Fraldas, Cerveja}
5	{Pão, Leite, Fraldas, Refrigerante}
...	...

Fonte: Elaborada pelo autor

- **Agrupamento:** Técnicas utilizadas para encontrar grupos de objetos (clusters) em que os elementos desse grupo têm características em comum, ou seja, que os elementos desse grupo possuem mais semelhança e correlação entre si do que com os elementos de outros grupos.

Figura 2.11: Agrupamento em Clusters



Fonte: Elaborada pelo autor

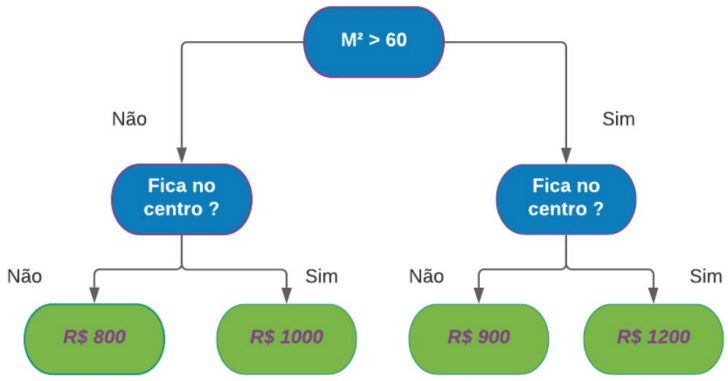
A Figura 2.11 ilustra que após aplicadas as técnicas de agrupamento nos dados originais, foram detectados 3 grupos, representados pelas cores amarelo, rosa e verde.

Tarefas Preditivas

Os modelos preditivos se referem a tarefa de construir o modelo para uma variável resposta que estará em função das variáveis ou atributos explicativos. Os dois tipos de tarefas serão apresentados a seguir.

- **Regressão:** Técnicas utilizadas para fazer previsão quantitativa para uma determinada variável com base nos valores de outras variáveis explicativas. Na Figura 2.12 temos um exemplo de problema de regressão onde se preve o valor do aluguel de um imóvel em função de outras variáveis como as medidas do espaço e a localidade.

Figura 2.12: Previsão de valor do aluguel



Fonte: Imagem extraída de medium.com. Fabio D. Junior [9]

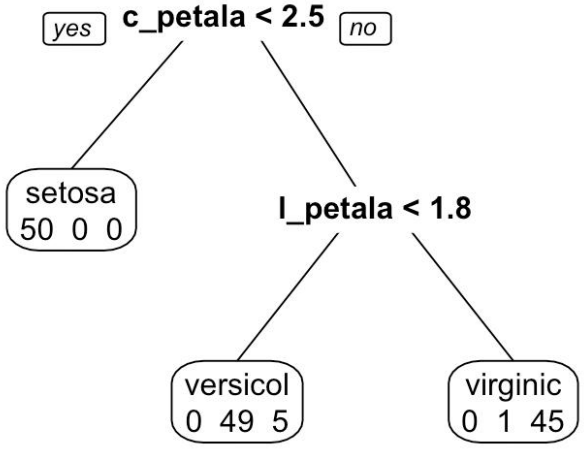
- **Classificação:** Esses métodos baseiam-se em prever a categoria de uma observação dada. Procura-se estimar um classificador que gere como saída a classificação qualitativa de um dado não observado em função dos valores dos outros atributos.

Figura 2.13: Flores de Íris

Fonte: Imagem extraída de wikipedia.com

Na Figura 2.13 são exibidos 3 tipos (classes) de flores: Setosa, Versicolor e Virgínica. A seguir ilustramos um exemplo em que medidas representando comprimento e largura da pétala de flores foram avaliadas para prever qual o tipo de cada flor.

Figura 2.14: Classificação em tipo de flor



Fonte: Elaborada pelo autor

Podemos observar na Figura 2.14 que algumas regras de associação foram criadas para analisar as variáveis e prever a classe. Desse modo, ao inserir uma nova observação de flor em que não sabemos a classe no modelo, as características dessa flor serão avaliadas considerando as regras de associação obtidas e ela será classificada como Setosa, Versicolor ou Virgínica.

Os métodos de classificação utilizam o chamado aprendizado de máquina supervisionado, onde a indução do modelo ocorre a partir de um conjunto de dados rotulados. Veremos adiante como funcionam as técnicas de aprendizado de máquina para classificação e aprofundaremos em um dos algoritmos mais populares para essa finalidade, chamado Árvore de Decisão.

2.2.1 Aprendizado de máquina

Aprendizado de máquina (em inglês, machine learning) é o termo utilizado para definir um conjunto de métodos estatísticos e computacionais para manipulação de dados que podem identificar padrões e tomar decisões de maneira automatizada.

Dentre os tipos mais populares de aprendizado de máquina, temos o aprendizado supervisionado, o não supervisionado, o semi-supervisionado, o aprendizado por reforço e as redes neurais. O foco do nosso trabalho será no supervisionado.

No método de aprendizado supervisionado, os algoritmos envolvidos são treinados por meio de exemplos rotulados da base de dados, ou seja, recebem um conjunto de variáveis explicativas junto com as variáveis respostas correspondentes e aprendem ao comparar os valores observados com os valores previstos. Para isso o conjunto de dados é dividido em treino e teste e o desempenho obtido pode ser calculado através de medidas de avaliação.

2.2.2 Treino e teste

Para problemas de classificação, é comum medir o desempenho de um classificador considerando a taxa de erro. O classificador prevê a classe de cada instância: se for correta, isso é contado como um sucesso, caso contrário é um erro. A taxa de erro é a proporção de erros

cometidos em todo um conjunto de instâncias e auxilia a medir o desempenho geral do classificador. O interesse central é o provável desempenho futuro em novos dados, não o desempenho passado em dados antigos. Já conhecemos as classificações de cada instância do conjunto de treinamento, e é por isso que podemos usá-lo para treinamento [7].

Embora não exista uma regra exata, o conjunto de treinamento normalmente abrange cerca de 75% das observações do banco de dados original [14]. Existem diferentes métodos para organização da base de dados em conjuntos de treinamento e teste. A seguir apresentamos mais detalhes sobre o bootstrap.

Bootstrap

O método de estimação boostrap é baseado no procedimento estatístico de amostragem com reposição, ou seja, um registro já escolhido para treinamento é colocado de volta no conjunto original para que tenha a mesma probabilidade de ser escolhido novamente. Se uma base de dados original possui n registros, uma amostra bootstrap terá em média 63.2% dos dados, como veremos a seguir.

A base de dados de tamanho n é amostrada n vezes, com substituição, para formar outro conjunto de n instâncias. Como alguns elementos neste segundo conjunto irão quase certamente ser repetidos, deve haver algumas instâncias no conjunto de dados original que não foram escolhidas, esses elementos serão utilizados como dados para teste [7].

Sabemos que a chance de sortear uma instância específica é de $\frac{1}{n}$, logo temos que a probabilidade de uma instância específica não ser escolhida para treinamento é de $1 - \frac{1}{n}$. Como faremos n sorteios, quando n é suficiente grande, a probabilidade se aproxima de:

$(1 - \frac{1}{n})^n \approx (e)^{-1} = 0.368$

Essa estrutura será essencial para a segunda parte do trabalho, onde será desenvolvida uma generalização do classificador Árvore de Decisão para Floresta Aleatória.

2.2.3 Avaliação de desempenho

Como dissemos anteriormente, o principal objetivo de um modelo é prever com sucesso o valor de saída para novos exemplos. Dentre as medidas mais utilizadas para essa função, temos a acurácia, sensibilidade, especificidade, taxa de vp e revocação. Cada medida possui sua peculiridade, portanto é importante utilizar mais de uma para diminuir a perda de informações. No nosso trabalho aprofundaremos na matriz de confusão ou tabela de contingência que é a base para várias medidas de erro e na acurácia.

Matriz de Confusão ou Tabela de Contingência:

A Tabela de Contingência permite calcular métricas que medem a capacidade discriminatória do modelo. Ela pode ser usada com duas ou mais classes para distringuir os tipos de erro. Veremos o que representam cada um dos seus elementos para um problema de 2 classes que pode ser visualizado na Tabela 2.1.

- VP (Verdadeiro Positivo): O número de instâncias observadas como positivas e previstas corretamente como positivas.

- FN (Falso Negativo): O número de instâncias observadas como positivas e previstas incorretamente como negativas.

- FP (Falso Positivo): O número de instâncias observadas como negativas e previstas incorretamente como positivas.

- VN (Verdadeiro Negativo): O número de instâncias observadas como negativas e previstas corretamente como negativas.

Tabela 2.1: Tabela de contingência

	Valor Observado		
Valor Previsto		Classe = 1	Classe = 0
	Classe = 1	VP	FP
	Classe = 0	FN	VN

Fonte: Elaborada pelo autor

Embora a matriz de confusão forneça as informações necessárias para determinar o desempenho do modelo de classificação, resumir essas informações em um único número auxilia na comparação do desempenho de diferentes modelos [12]. Isso pode ser feito usando uma métrica de desempenho como a acurácia. A acurácia é a proporção de predições corretas em todas as previsões. Essa medida pode ser obtida através da seguinte fórmula:

$$\text{Acurácia} = \frac{\text{Número de previsões corretas}}{\text{Número total de previsões}} = \frac{\text{VP+VN}}{\text{VP+VN+FP+FN}}$$

Analogamente, a performance do modelo pode ser medida através da taxa de erro, que é obtida pela seguinte equação:

$$\text{Taxa de erro} = \frac{\text{Número de previsões incorretas}}{\text{Número total de previsões}} = \frac{\text{FP + FN}}{\text{VP+VN+FP+FN}}$$

Outras estatísticas úteis podem ser obtidas pela tabela de contingência, como é o caso da sensibilidade e especificidade, que medem respectivamente a taxa de verdadeiros positivos e a taxa de verdadeiros negativos.

$$\text{Sensibilidade} = \frac{\text{Número de previsões positivas corretas}}{\text{Número total de previsões positivas}} = \frac{\text{VP}}{\text{VP+FN}}$$

$$\text{Especificidade} = \frac{\text{Número de previsões negativas corretas}}{\text{Número total de previsões negativas}} = \frac{\text{VN}}{\text{VN+FP}}$$

2.3 Árvore de Decisão

Árvore de decisão é o nome dado a um dos métodos estatísticos mais populares para tarefas de classificação e previsão de dados. Nesse método, um problema complexo é decomposto em subproblemas mais simples (estratégia dividir para conquistar) [12]. A raiz, os nós, as ramificações e as folhas da árvore têm uma representação particular dentro do algoritmo que será apresentado.

Cada nó interno testa um atributo com uma condição geralmente constante, cada ramo (galho) ou aresta corresponde a um valor do atributo que será testado e cada folha representa uma classe. A classe da folha é rotulada como uma função e considera os valores da variável alvo dos exemplos. O nó da raiz não possui aresta de entrada, apenas de saída. Resumidamente podemos dizer que o método estatístico de árvore de decisão utiliza das respostas obtidas pelos "nós filhos" para conduzir a regra da função que armazena os dados em seu nó raiz.

Um exemplo simples de aplicação desse método pode ser visualizado na Figura 2.15 a seguir. Deseja-se fazer a previsão de ir ou não ir à praia considerando as respostas dos nós filhos após testar as condições através dos ramos.

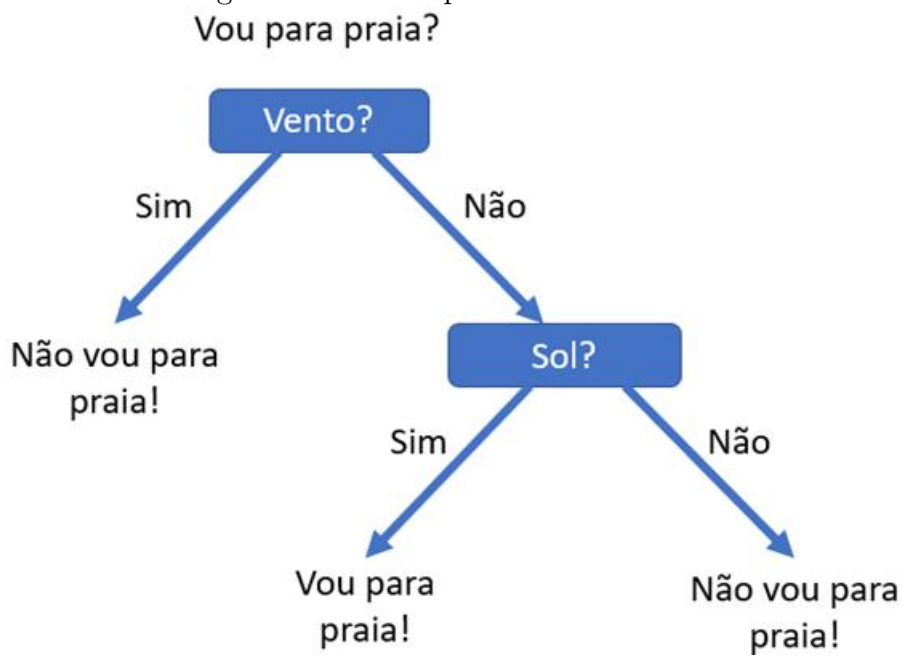
Figura 2.15: Matriz de decisão

Dia	Sol?	Vento?	Vou para praia?
1	Sim	Sim	Não
2	Sim	Sim	Não
3	Sim	Não	Sim
4	Não	Não	Não
5	Não	Sim	Não
6	Não	Sim	Não

Fonte: Imagem extraída de didatica.tech [19]

Percebemos que no primeiro teste para o nó do atributo "Vento?", todas as observações que tiveram valor nominal "Sim" correspondem a classe "Não" para ir a praia, os demais atributos foram testados seguindo a mesma lógica. O fim da árvore ocorre quando não se tem mais divisão e as classes foram previstas. A Figura 2.16 ilustra a estrutura dessa árvore de decisão gerada.

Figura 2.16: Exemplo árvore de decisão



Fonte: Imagem extraída de didatica.tech [19]

Pode existir mais de uma árvore que se adequa aos dados, começando a testar as condições a partir de outros atributos. Por exemplo, se começarmos a testar a árvore pelo atributo "Sol?", veremos que todos os dias que não fizeram sol não se foi a praia. Assim como podemos ter mais de uma árvore para um mesmo problema, também podemos utilizar diferentes métodos de cálculo para a seleção dos melhores critérios de divisão.

No método de árvore de decisão para obter resultados de classificação, algumas medidas são utilizadas a fim de medir a heterogeneidade dos dados e definir critérios para seleção de atributos. Apresentaremos na sessao 2.3.1 o índice Gini (que será aplicado na nossa base de dados posteriormente) e a entropia.

2.3.1 Seleção das variáveis para divisão

Existem muitas medidas que podem ser usadas para determinar a melhor maneira de divisão para os nós da árvore. Essas medidas são definidas em termos da distribuição de classe das observações antes e depois da divisão [12].

Gini

O índice Gini, também conhecido por coeficiente ou razão de Gini, é uma medida desenvolvida pelo estatístico italiano Corrado Gini em 1912. Nos algoritmos de Árvore de Decisão, esse índice de dispersão estatística nos auxilia a medir a heterogeneidade dos dados e a selecionar os atributos. Para um conjunto de dados contendo várias observações, cada obvervação com

uma classe i, a equação do índice Gini é definida pela seguinte expressão:

$$\text{Gini}(t) = 1 - \sum_{i=0}^{c-1} p(i|t)^2$$

Em que c equivale ao número de classes e $p(i|t)$ representa a fração de observações pertencentes a classe i em um determinado nó t.

O índice de Gini assume um valor próximo de zero se um nó conter predominantemente observações de instâncias pertencentes a mesma classe e se afasta de zero quando o conjunto apresenta as observações distribuidas igualmente entre todas as classes. Uma alternativa ao índice de Gini é a entropia.

ENTROPIA

No contexto das árvores de decisão para classificação, a entropia também é usada para avaliar a qualidade de uma divisão específica em cada nó. Uma forma de se obter essa medida é através da equação:

$$\text{Entropia}(t) = -\sum_{i=0}^{c-1} p(i|t)\log_2 p(i|t)$$

Como $0 \leq p(i|t) \leq 1$, segue que $0 \leq -p(i|t)\log_2 p(i|t)$. Podemos perceber que a entropia assumirá um valor próximo de um em um conjunto de dados heterogêneo e próximo de zero se um nó conter observações distribuidas igualmente entre todas as classes.

2.3.2 OVERFITTING (SOBREAJUSTE)

Quando um determinado método produz um bom desempenho nos dados de treino mas baixa capacidade preditiva para os dados de testes, chamamos esse fenômeno de overfitting. Isso acontece porque os procedimentos estatísticos para aprendizado tentam encontrar padrões no conjunto de treinamento de modo muito rigoroso, com isso pode estar associando alguns padrões que são causados apenas pelo acaso ao invés de propriedades verdadeiras do comportamento dos nossos dados. Desse modo, ao sobrepormos os dados de treinamento, o erro do teste será grande pois os supostos padrões que o método encontrou nos dados de treino não existem nos dados do teste [5].

Analogamente, o underfitting ocorre quando o algoritmo não encontra uma boa relação entre os dados de treinamento. Desse modo o modelo apresenta baixa utilidade e o teste pode ser dispensado.

A fim de prevenir o processo de sobreajuste é recomendado definir parâmetros para o tamanho da árvore, como o valor mínimo de amostra para a divisão de nós e para cada folha, o nível de profundidade da árvore e o valor máximo de atributos para cada divisão. Além da definição dessas restrições, podemos aplicar um método denominado processo de podagem, produzindo uma árvore final menos complexa e com melhor capacidade preditiva.

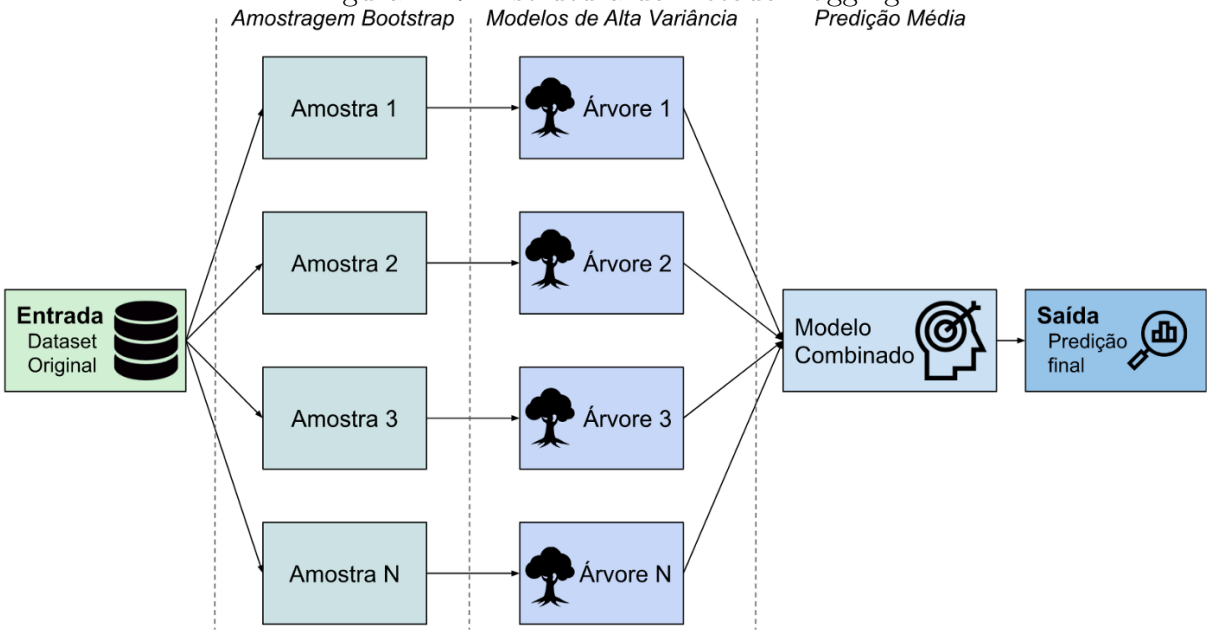
2.4 FLORESTA ALEATÓRIA

O algoritmo de Floresta Aleatória, como o próprio nome sugere, tem em sua composição várias árvores de decisão. A ideia básica é fazer a previsão para diversas árvores, combiná-las e obter um único modelo. Com o modelo criado, após apresentar novos dados, cada árvore criada irá exibir o seu resultado, sendo que em problemas de regressão será realizada a média dos valores previstos, e esta média informada como resultado final, e em problemas de classificação o resultado que for mais abundante na votação individual dos estimadores será o escolhido. As técnicas que são utilizadas nesse método podem melhorar a capacidade de generalização do modelo e consequentemente aumentar a capacidade preditiva, evitando overfitting [5].

2.4.1 BOOTSTRAP AGGREGATION (BAGGING)

Nas etapas de divisão das bases para a criação das árvores que integram a floresta, utilizamos o método bootstrap apresentado em 2.2.2. Uma das metas do bootstrap é obter vantagem sobre a variação de amostragem. Nesse caso, são feitas várias amostragens e em cada uma ocorre uma reposição interna das instâncias, de modo que as bases de treino geradas não sejam idênticas e as bases de teste tenham mais variabilidade. O método que utiliza a combinação de bootstrap com árvore de decisão é chamado de Bagging [5]. A Figura 2.17 ilustra esta estrutura.

Figura 2.17: Estrutura do método Bagging



Fonte: Imagem extraída de medium.com [18]

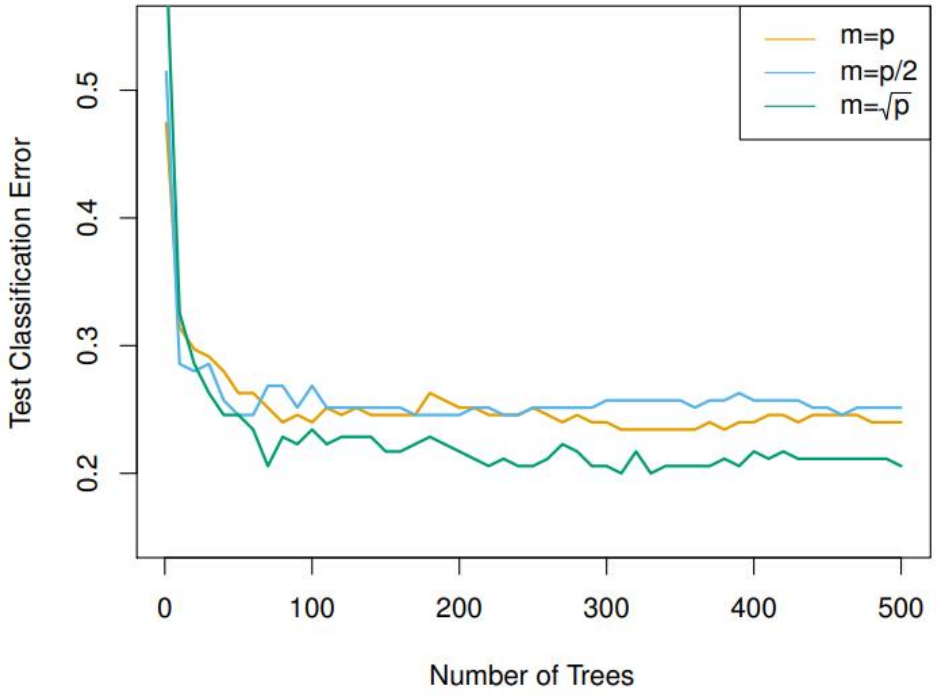
Quando utilizamos o bootstrap em uma base de dados, uma amostra não conterá todos as observações originais do treino. Os registros que não serão utilizados nessa amostragem de treino são chamados de Out-of-bag (OOB) e podem ser utilizados para teste. A medida que aumentamos a quantidade de reamostragens por bootstrap, aumenta-se a chance de que cada observação seja utilizada para o caso de teste em alguma árvore do modelo final.

Como foi demonstrado em 2.2.2, cada observação aparecerá em média em $\frac{2}{3}$ das árvores e uma vez que a resposta para cada observação é prevista usando apenas as árvores que não foram ajustadas usando essa observação, o erro OOB resultante é uma medida válida para avaliarmos o erro do nosso teste preditivo [5].

2.4.2 SELEÇÃO DAS VARIÁVEIS PARA DIVISÃO

Em Floresta Aleatória, o método de seleção das variáveis que serão consideradas em cada divisão da árvore ocorre um pouco diferente do que foi apresentado em 2.3.1. O algoritmo selecionará de maneira aleatória m variáveis para serem avaliadas em cada nó e as demais não serão incluídas nessa divisão, onde $m = \sqrt{p}$ e p representa a quantidade total de variáveis da nossa base. O fato de variáveis que tem alto poder preditivo serem removidas em uma única divisão afetará a capacidade de previsão dessa árvore, mas como serão construídas várias árvores, esse erro será superado e podemos diminuir o overfitting. Provavelmente as árvores seguintes serão diferentes da primeira, pois os processos de seleção de amostra e de variáveis ocorreram de maneira aleatória.

Figura 2.18: Resultados de Floresta Aleatória aplicados em uma base com 500 variáveis



Fonte: An Introduction to Statistical Learning with Applications in R [5]

Na Figura 2.18, cada linha colorida corresponde a um valor diferente de m, o número de variáveis disponíveis para cada divisão dos nós da árvore. Podemos observar que $m = \sqrt{p}$ representado pela linha verde obteve menor taxa de erro.

3. METODOLOGIA

3.1 MATERIAIS E MÉTODOS

A fim de realizar técnicas para reconhecimento de padrões e classificação dos oponentes, a nossa primeira tarefa necessária foi obter uma base de dados de jogos reais (entre jogadores humanos) e que seja pública. Há uma grande quantidade de dados de jogos disponíveis na web, mas a maioria é de propriedade de cassinos online e, portanto, não são gratuitos. Neste ponto, gostaríamos de agradecer a David de HandHQ.com [3] que nos forneceu 22 milhões de registros de partidas manipulados para fins acadêmicos e analíticos. Esses dados se referem a partidas disputadas entre abril de 2009 e agosto de 2010 (um período de 16 meses). Os registros tiveram o nome da mesa, o ID da mão e os nomes dos jogadores alterados com o intuito de preservar o anonimato.

Além disso, realizamos alguns procedimentos a fim de transformar esses registros para o formato de base de dados que desejamos. Essas etapas serão apresentadas e analisadas a seguir.

3.1.1 PRÉ-PROCESSAMENTO DOS DADOS

Os registros da base de dados original obtida variam de site para site mas todas seguem uma estrutura bem similar, que pode ser visualizada na Figura 3.1. Em cada registro estão contidas as informações de como ocorreu uma determinada partida, do início ao fim. Dentre elas temos a posição, o nome e o montante de fichas inicial (stack) de cada jogador na mesa, as ações que ocorreram em cada rodada (incluindo o valor das apostas), as cartas comunitárias expostas e o resultado final.

Figura 3.1: Exemplo de uma observação da base de dados [3]

```
Stage #3017637534: Holdem (1 on 1)  No Limit $10 - 2009-07-01 02:17:57 (ET)
Table: ALPS AVE (Real Money) Seat #6 is the dealer
Seat 6 - rUUVdCewx1gNEYPR2OimZA ($1,529 in chips)
Seat 4 - uJcc0SNtGLr5m8bORzF8FA ($1,600.52 in chips)
rUUVdCewx1gNEYPR2OimZA - Posts small blind $5
uJcc0SNtGLr5m8bORzF8FA - Posts big blind $10
*** POCKET CARDS ***
rUUVdCewx1gNEYPR2OimZA - Raises $25 to $30
uJcc0SNtGLr5m8bORzF8FA - Calls $20
*** FLOP *** [9d 3d 5h]
uJcc0SNtGLr5m8bORzF8FA - Checks
rUUVdCewx1gNEYPR2OimZA - Bets $48
uJcc0SNtGLr5m8bORzF8FA - Calls $48
*** TURN *** [9d 3d 5h] [2c]
uJcc0SNtGLr5m8bORzF8FA - Checks
rUUVdCewx1gNEYPR2OimZA - Checks
*** RIVER *** [9d 3d 5h 2c] [7s]
uJcc0SNtGLr5m8bORzF8FA - Bets $110
rUUVdCewx1gNEYPR2OimZA - Calls $110
*** SHOW DOWN ***
uJcc0SNtGLr5m8bORzF8FA - Shows [9c Qd] (One pair, nines)
rUUVdCewx1gNEYPR2OimZA - Mucks
uJcc0SNtGLr5m8bORzF8FA Collects $375.25 from main pot
*** SUMMARY ***
Total Pot($376) | Rake ($0.75)
Board [9d 3d 5h 2c 7s]
Seat 4: uJcc0SNtGLr5m8bORzF8FA (big blind) won Total ($375.25)
HI:($375.25) with One pair, nines [9c Qd - B:9d,P:9c,P:Qd,B:7s,B:5h]
Seat 6: rUUVdCewx1gNEYPR2OimZA (dealer) (small blind) HI: [Mucked] [5d Kd]
```

Fonte: Elaborada pelo autor.

Existem softwares que foram desenvolvidos para importar os dados nesse formato e que oferecem diversas funções para o usuário. Uma dessas funções que temos interesse para o nosso trabalho é a de utilizar filtros para visualizarmos estatísticas descritivas a respeito da amostra. Utilizamos o software Hold'em Manager. As instruções básicas para instalação dessa ferramenta encontram-se no link na nota desta página[1].

Após a importação da base, podemos visualizar na sessão "Opponents" uma lista dos jogadores e colunas que representam variáveis indicando frequências médias de ações realizadas, quantidade de partidas disputadas e o lucro ou prejuízo obtido. Nessa sessão é possível utilizar alguns filtros para manipulação da base.

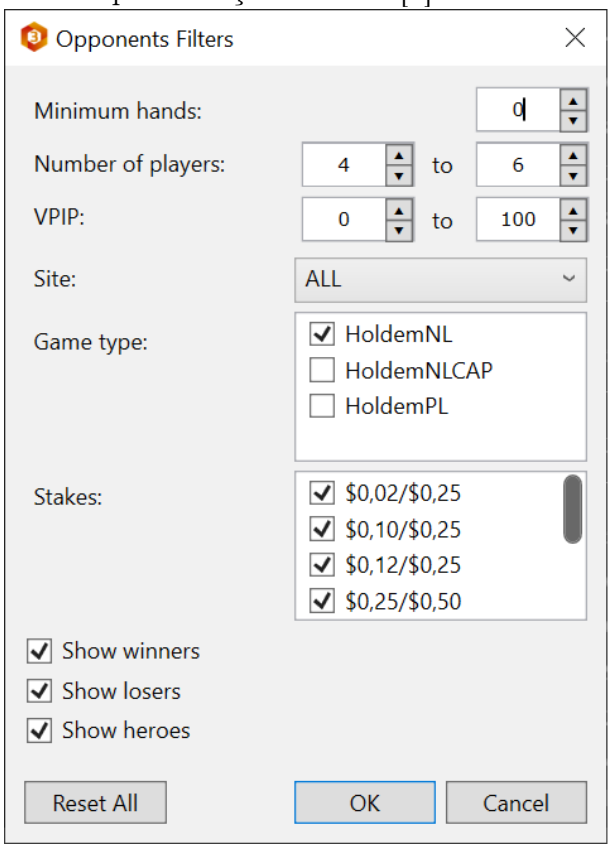
Inicialmente consideramos utilizar um filtro para que a base contenha apenas jogadores que tiveram mais de 10.000 partidas disputadas. Nossa motivação inicial para analisar apenas

[1] https://kb.holdemmanager.com/knowledge-base/article/setup-wizard-advanced-install (acessado em 15/03/2022)

esses jogadores é que algumas variáveis a serem analisadas precisam de oportunidades para ocorrer e convergir, portanto quando o jogador tem poucas partidas disputadas a variabilidade dessas frequências é alta e ocorrem muitos casos em que não se teve oportunidade para tomar determinada ação, gerando muitos valores ausentes na base.

No entanto, ao selecionarmos apenas jogadores com maior volume de partidas, temos uma perda significativa de observações contendo indivíduos que tiveram prejuízo. Acreditamos que uma justificativa razoável para essa perda é que os jogadores mais habilidosos tendem a participar mais, enquanto os menos habilidosos têm um volume menor de jogo. Como é de nosso interesse ter instâncias contendo ambas as classes (perdedor e vencedor), optamos por não utilizar o filtro de quantidade de partidas e manter todas as observações de jogadores. Os filtros finais aplicados podem ser verificados na Figura 3.2.

Figura 3.2: Filtros para seleção da base [3] no Holdem Manager [6]

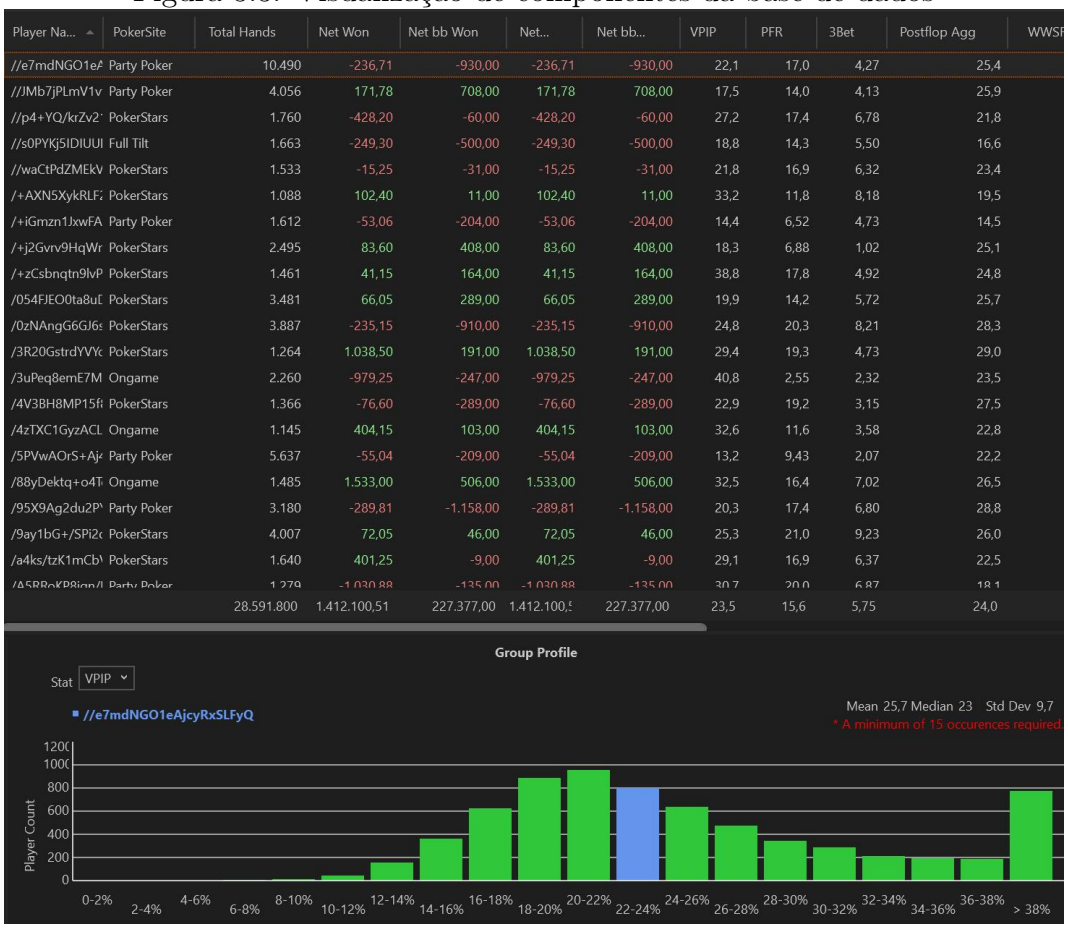


Fonte: Elaborada pelo autor. Base de dados [3] manipulada no software Holdem Manager [6]

3.2 SELEÇÃO DE ATRIBUTOS

Na Figura 3.3 a seguir exibimos algumas colunas geradas e o que representa cada uma das principais variáveis. Algumas delas serão utilizadas nos modelos de previsão considerados neste trabalho.

Figura 3.3: Visualização de componentes da base de dados



Player Na...	PokerSite	Total Hands	Net Won	Net bb Won	Net...	Net bb...	VPIP	PFR	3Bet	Postflop Agg	WWSF
//e7mdNGO1eA	Party Poker	10.490	-236,71	-930,00	-236,71	-930,00	22,1	17,0	4,27	25,4	
//JMb7jPLmV1v	Party Poker	4.056	171,78	708,00	171,78	708,00	17,5	14,0	4,13	25,9	
//p4+YQ/krZv2	PokerStars	1.760	-428,20	-60,00	-428,20	-60,00	27,2	17,4	6,78	21,8	
//s0PYKj5IDIUUI	Full Tilt	1.663	-249,30	-500,00	-249,30	-500,00	18,8	14,3	5,50	16,6	
//waCtPdZMEkV	PokerStars	1.533	-15,25	-31,00	-15,25	-31,00	21,8	16,9	6,32	23,4	
/+AXN5XykRLF	PokerStars	1.088	102,40	11,00	102,40	11,00	33,2	11,8	8,18	19,5	
/+iGmzn1JxwFA	Party Poker	1.612	-53,06	-204,00	-53,06	-204,00	14,4	6,52	4,73	14,5	
/+j2Gvrv9HqWr	PokerStars	2.495	83,60	408,00	83,60	408,00	18,3	6,88	1,02	25,1	
/+zCsbnqtn9lvP	PokerStars	1.461	41,15	164,00	41,15	164,00	38,8	17,8	4,92	24,8	
/054FJEO0ta8uI	PokerStars	3.481	66,05	289,00	66,05	289,00	19,9	14,2	5,72	25,7	
/0zNAngG6GJ6	PokerStars	3.887	-235,15	-910,00	-235,15	-910,00	24,8	20,3	8,21	28,3	
/3R20GstrdYVY	PokerStars	1.264	1.038,50	191,00	1.038,50	191,00	29,4	19,3	4,73	29,0	
/3uPeq8emE7M	Ongame	2.260	-979,25	-247,00	-979,25	-247,00	40,8	2,55	2,32	23,5	
/4V3BH8MP15f	PokerStars	1.366	-76,60	-289,00	-76,60	-289,00	22,9	19,2	3,15	27,5	
/4zTXC1GyzACL	Ongame	1.145	404,15	103,00	404,15	103,00	32,6	11,6	3,58	22,8	
/5PVwAOrS+Aj	Party Poker	5.637	-55,04	-209,00	-55,04	-209,00	13,2	9,43	2,07	22,2	
/88yDektq+o4T	Ongame	1.485	1.533,00	506,00	1.533,00	506,00	32,5	16,4	7,02	26,5	
/95X9Ag2du2P	Party Poker	3.180	-289,81	-1.158,00	-289,81	-1.158,00	20,3	17,4	6,80	28,8	
/9ay1bG+/SPi2	PokerStars	4.007	72,05	46,00	72,05	46,00	25,3	21,0	9,23	26,0	
/a4ks/tzK1mCb	PokerStars	1.640	401,25	-9,00	401,25	-9,00	29,1	16,9	6,37	22,5	
/A5RRoKP8ign/I	Party Poker	1.279	-1.030,88	-135,00	-1.030,88	-135,00	30,7	20,0	6,87	18,1	
		28.591.800	1.412.100,51	227.377,00	1.412.100,5	227.377,00	23,5	15,6	5,75	24,0	

Fonte: Elaborada pelo autor.

- **Player Name:** Nome do jogador. Como dissemos anteriormente, esses nomes foram alterados para que a base possa ser utilizada em pesquisas acadêmicas sem complicações burocráticas.

- **Total Hands:** O número de mãos jogadas por cada invidíduo.

- **Net Won, Net bb Won, Net Won Hand e Net bb Won Hand:** Essas variáveis representam o valor do saldo, positivo ou negativo, do jogador. Selecionaremos para nosso modelo "Net bb Won"para ser a classe que vamos prever posteriormente. O diferencial desejado da variável escolhida é que ela tem capacidade de generalização por indicar o saldo final obtido na unidade de bigblinds. Desse modo o saldo atingido por jogadores em mesas de valores iniciais diferentes recebem o mesmo peso.

- **VPIP:** "Voluntarily Put Money in Pot". Essa estatística revela a média das vezes que o

jogador optou por participar de uma partida após receber suas cartas na etapa inicial do jogo (pré-flop). A fórmula para cálculo do VPIP é:

$$\text{VPIP} = \frac{\text{Total das vezes que voluntariamente optou por participar pré-flop}}{\text{Total das oportunidades para participar pré-flop}} \text{ x } 100$$

- **PFR:** "Pré-Flop Raise" é similar ao VPIP. Indica a média das vezes que o jogador optou por participar de uma partida agressivamente após receber suas cartas no início do jogo, ou seja, a ação escolhida foi de raise (aumentar). A fórmula para cálculo do PFR é:

$$\text{PFR} = \frac{\text{Total das vezes que voluntariamente aumentou a aposta pré-flop}}{\text{Total das mãos jogadas pré-flop}} \text{ x } 100$$

- **3bet:** Mostra a porcentagem das vezes que um jogador reaumentou uma aposta na etapa inicial do jogo, previamente houve uma aposta e um aumento (2bet). A fórmula para cálculo da 3bet é:

$$\text{3bet} = \frac{\text{Total das vezes que reaumentou uma aposta pré-flop}}{\text{Total de oportunidades para reaumentar pré-flop}} \text{ x } 100$$

- **Fold to 3bet:** Porcentagem das vezes que um jogador aumentou uma aposta na etapa inicial do jogo e desistiu para um reaumento (3bet). A fórmula para cálculo do Fold para 3bet é:

$$\text{Fold to 3bet} = \frac{\text{Total das vezes que desistiu contra uma 3bet pré-flop}}{\text{Total de oportunidades de desistir contra 3bet pré-flop}} \text{ x } 100$$

- **Postflop Agg:** Fator de agressão, as vezes chamado "Agg%", é uma estatística relacionada a frequências de movimentos após a abertura das cartas comunitárias, ela representa um indicativo do nível de agressividade dos jogadores no postflop. Por exemplo, se um jogador aposta no flop e no turn mas desiste ou passa a vez no river, seu Agg% é 0,66% porque ele fez 2 de 3 ações agressivas durante cada rodada. Bet, raise e check-raise são considerados ações agressivas, enquanto check e fold não. A fórmula para cálculo de Postflop Agg é:

$$\text{Postflop Agg} = \frac{\text{Número de ações agressivas no postflop}}{\text{Número de streets vistas}} \text{ x } 100$$

- **WTSD:** A média de "Went to Showdown" indica a porcentagem de vezes que o jogador foi para o showdown após chegar no flop. Ou seja, participou da mão até o final, chegando a revelar suas cartas. A fórmula para cálculo do WTSD é:

$$\text{WTSD} = \frac{\text{Total das vezes que foi para o showdown}}{\text{Total das mãos que viu o flop}} \text{ x } 100$$

- **WWSD e Won SD:** "Win When Saw Flop" e "Won at Showdown" representam res-

pectivamente a porcentagem de vezes que o jogador ganhou a partida após chegar no flop e a porcentagem de vezes que ele ganhou quando participou até o momento de revelar suas cartas. Optamos por não considerar essas variáveis para fazer a previsão da nossa classe pois elas ja são indicadores de frequências de vitória e nosso interesse principal está em frequências de ações realizadas.

- **Outras variáveis:** 17 outras variáveis que representam frequências de ações também foram disponibilizadas, sendo elas: "Flop Cbet, Turn Cbet, River Cbet, Fold to Flop Cbet, Fold to Turn Cbet, Fold to River Cbet, Raise to Flop Cbet, Raise to Turn Cbet, Raise to River Cbet, Squeeze, Call Two Raisers, Raise Two Raisers, Call vs 3bet, Raise vs 3bet, Fold to 4bet, Call to 4bet e Raise to 4bet".

Essas variáveis representam situações mais improváveis de ocorrer do que as descritas anteriormente. Portanto, como os jogadores tiveram menos oportunidades para realizar ações nessas situações, temos muitas observações com valores ausentes nessas colunas.

Os dados estão dispostos do modo que queremos para prosseguir com os métodos de aprendizado de máquina no software R [15]. Exportaremos essa nova base de dados gerada pelas colunas para um arquivo no formato .csv.

3.2.1 VALORES AUSENTES (NAS)

Após entrarmos com os dados no software R, detectamos e removemos variáveis em que os valores ausentes ultrapassaram 60%, dessa forma preservamos observações que podem afetar a capacidade preditiva do nosso modelo. Também não incluímos o nome dos jogadores e o total de partidas pois nosso interesse é de relacionar as variáveis que representam frequências de ações. A Figura 3.4 revela parte da composição da base após essas modificações.

Figura 3.4: Visualização de componentes da base de dados

```
'data.frame':   176419 obs. of  14 variables:
 $ NetbbWon        : num  183 -123 4001 -510 -111 ...
 $ VPIP            : num  0.212 0.233 0.283 0.244 0.2ﾒ ...
 $ PFR             : num  0.175 0.158 0.235 0.111 0.153 ...
 $ X3Bet           : num  0.0574 0.0407 0.0824 0.0459 ...
 $ PostflopAgg     : num  0.265 0.179 0.282 0.175 0.21 ...
 $ WTSD            : num  0.238 0.305 0.311 0.29 0.276 ...
 $ FlopCBet        : num  0.654 0.516 0.605 0.615 0.567 ...
 $ TurnCBet        : num  0.4 0.489 0.502 0.503 0.492 ...
 $ Foldvs.FlopCBet : num  0.544 0.511 0.378 0.649 0.554 ...
 $ Raisevs.FlopCBet: num  0.1159 0.156 0.1543 0.0561 ...
 $ Squeeze         : num  0.0582 0.0353 0.0581 0.0349 ...
 $ Foldto3Bet      : num  0.681 0.546 0.597 0.412 0.473 ...
 $ Callvs.3Bet     : num  0.237 0.389 0.276 0.505 0.423 ...
 $ Raisevs.3Bet    : num  0.0825 0.0652 0.1273 0.0824 ...
```

Fonte: Elaborada pelo autor

Em seguida criamos uma função para percorrer toda a base e remover as observações que ainda continham valores ausentes. Através desse procedimento geramos um novo conjunto

contendo 46112 observações. As linhas de código que foram utilizadas podem ser verificadas no Apêndice Código R deste trabalho.

3.3 Definição do grupo de classes

Temos interesse em obter grupos de classes que contenham uma proporção aproximada de registros contendo jogadores perdedores e vencedores, ou seja, grupos heterogêneos em relação a variável saldo. Desse modo os algoritmos utilizados têm a oportunidade de aprender regras de associação e prever o comportamento de ambos os tipos de jogadores sem predominância de uma classe sobre a outra.

Criamos uma função para consultar todas as observações da base avaliando a variável quantitativa "Net Won bb" e separando os jogadores em três grupos distintos. Sendo eles: perdedor, empatado e vencedor. O intervalo final utilizado para separar as classes foi: perdedor $< -650 <$ empatado $< 700 <$ vencedor. Ou seja, se o jogador teve um saldo final inferior a -650 big blinds, ele foi categorizado como perdedor, se seu saldo final foi superior a 700 big blinds ele foi categorizado como vencedor; caso contrário o jogador foi categorizado como empatado.

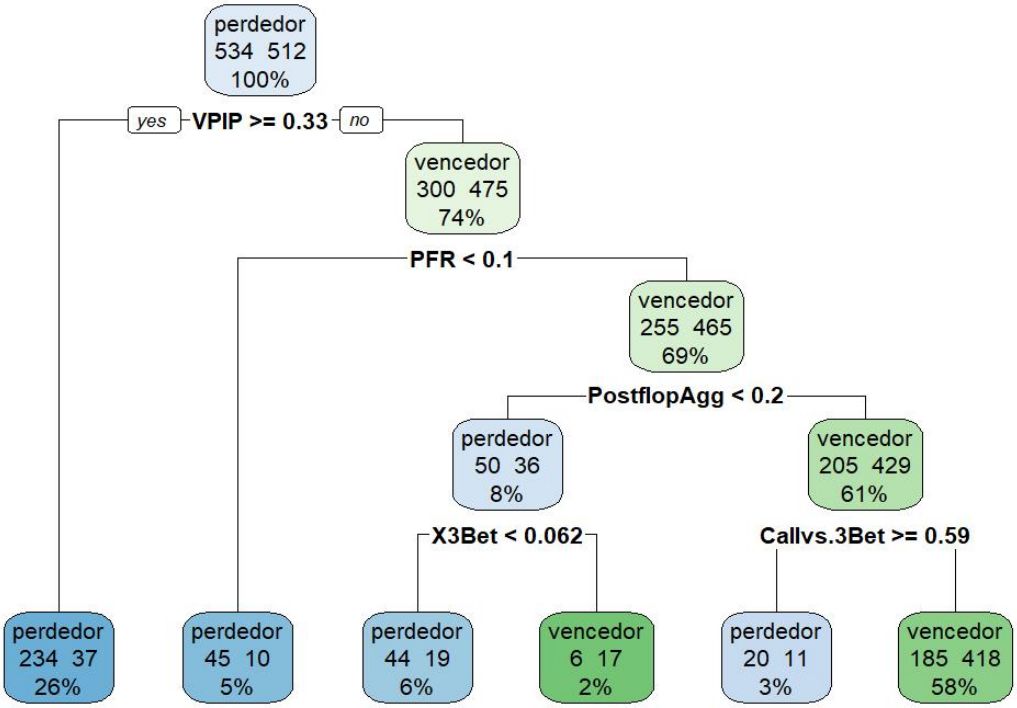
Removemos as observações dos jogadores que definimos como empatados e a base final obtida conteve 1308 jogadores, sendo 659 perdedores e 649 vencedores. Com as classes definidas, podemos aplicar os algoritmos de Árvore de Decisão e Floresta Aleatória. Para isso utilizaremos os pacotes rpart, rpart.plot e randomForest [15].

4. RESULTADOS

4.1 UTILIZANDO ÁRVORE DE DECISÃO

Primeiramente dividimos a base de maneira aleatória em 80% para treino e 20% para teste e consideramos todas as 13 variáveis preditoras para criar regras de associação. O índice Gini foi utilizado como medida para seleção da melhor divisão em cada nó. Alteramos a profundidade máxima da árvore para 6 nós de divisões para obter uma melhor visualização das regras de associação e evitar o sobreajuste. As demais configurações de parâmetros do algoritmo seguiram o padrão do software R.

Figura 4.1: Modelo de árvore de decisão



Fonte: Elaborada pelo autor

Através da árvore gerada que foi exibida na Figura 4.1, podemos constatar as regras de associação produzidas e a estrutura da árvore. No topo da árvore, verificamos que para esse modelo utilizando o conjunto de treinamento, temos 534 observações de jogadores perdedores e 512 de vencedores, o nome da classe predominante no momento de cada divisão é indicado acima desses números. O nó raiz, por exemplo, indica uma regra de associação em que jo-

gadores que apresentaram a frequência média da variável VPIP (porcentagem de vezes que optou por participar de uma partida após receber suas cartas iniciais) maior que 33% foram observados como perdedores, essas observações correspondem a 26% dos dados reservados para treinamento, sendo que esses 26% representam 234 jogadores perdedores e 37 vencedores.

Outra sequência de divisões de nós que representa a maior parte das observações (58%), pode ser verificada ao examinar o lado direito da árvore. Nesse caso, dos jogadores que na base de treino apresentaram VPIP menor que 33%, PFR maior que 10%, PostFlopAgg maior que 20% e Callvs.3Bet menor ou igual a 59%, foram classificados como vencedores. Essas regras de associação sugerem um perfil de jogador que, se comparado as observações dos perdedores, participa de menos partidas e de modo mais agressivo, tanto nas etapas iniciais do jogo como nas rodadas posteriores (PostFlop), tendo maior predominância de ações em que efetua ou aumenta uma aposta.

As medidas de desempenho gerais do modelo podem ser extraidas após a criação das regras de associação no treinamento e aplicação no conjunto de teste. Utilizamos o aprendizado para prever a classe dos 262 jogadores reservados à base de teste e a matriz de confusão com os resultados obtidos pode ser visualizada na Tabela 4.1.

Tabela 4.1: Tabela de contingência para as previsões do conjunto teste

	Observado		
Previsto		Perdedor	Vencedor
	Perdedor	71	22
	Vencedor	54	115

Fonte: Elaborada pelo autor

A acurácia total observada para esse modelo foi de aproximadamente 71%, enquanto a sensibilidade (classe perdedora classificada corretamente) foi de 56,8% e a especificidade (jogadores vencedores classificados corretamente) de 83,94%.

4.2 Utilizando uma Floresta Aleatória

A fim de fazer a previsão para diversas árvores, combiná-las e obter um único modelo, utilizamos o pacote randomForest [15]. No nosso modelo de classificação, configuramos os parâmetros do algoritmo para considerar 4 variáveis em cada divisão dos nós por árvore ("mtry=4"), como temos 13 variáveis preditoras ($p = 13$), utilizaremos $m = \sqrt{p}$ conforme sugerido na Figura 2.18, que arredondando para o inteiro mais próximo resulta em 4. Para evitar o sobreajuste, consideramos 15 o número máximo de nós terminais ("maxnodes") para cada árvore. Adicionamos o parâmetro "importance=TRUE" a fim de visualizar o desempenho de cada variável dentro do modelo. As demais configurações de parâmetros do algoritmo seguiram o padrão do software R.

Figura 4.2: Resultados da Floresta Aleatória considerando 500 árvores

```
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 4

        OOB estimate of  error rate: 29.83%
Confusion matrix:
         perdedor vencedor class.error
perdedor      293      241   0.4513109
vencedor       71      441   0.1386719
> importance(floresta.poker)
                  perdedor  vencedor MeanDecreaseAccuracy MeanDecreaseGini
VPIP             8.7807536 28.100520            30.367826        48.115078
PFR              0.6624696 15.664430            15.615319        11.417493
X3Bet            3.1547164  3.406501             5.952202         3.964619
PostflopAgg      6.0794886  5.505472             8.086836         7.464195
WTSD             9.1588791  3.292736            10.980796         7.402190
FlopCBet        -5.1128725  4.013614            -1.164704         2.993642
TurnCBet        -3.3082954  6.265047             4.117014         6.312913
Foldvs.FlopCBet  0.7263440  3.182563             3.336011         4.298696
Raisevs.FlopCBet 3.1095882 -1.189189             1.816796         2.654799
Squeeze         -2.1042507  8.032734             7.504251         6.322959
Foldto3Bet       6.8963292 12.656762            16.092166        26.183053
Callvs.3Bet      6.9960719 18.612941            23.174512        35.805001
Raisevs.3Bet    -7.9365338 10.883882             8.264352         6.969812
```

Fonte: Elaborada pelo autor

Considerando os parâmetros citados, onde o padrão do algoritmo gera 500 árvores para o modelo, os resultados visualizados na Figura 4.2 indicam uma estimativa de erro para os OOB de 29,83% na base de treinamento. Ao verificar a importância de cada variável, vemos na coluna "MeanDecreaseAccuracyque FlopCbet pode estar causando um aumento do erro, ou seja, contribui de forma negativa para a acurácia. Ainda assim a capacidade preditiva do modelo aplicado no conjunto de teste foi superior ao caso em que consideramos apenas uma árvore, como pode ser visualizado na Tabela 4.2.

Tabela 4.2: Tabela de contingência para a Floresta Aleatória considerando 500 árvores

	Observado		
Previsto		Perdedor	Vencedor
	Perdedor	72	16
	Vencedor	53	121

Fonte: Elaborada pelo autor

A acurácia total observada para esse modelo foi de aproximadamente 73,66%, enquanto a sensibilidade (classe perdedora classificada corretamente) foi de 57,6% e a especificidade (jogadores vencedores classificados corretamente) de 88,32%.

Alteramos o parâmetro "ntree" do algoritmo para considerar 10.000 árvores a fim de verificar se a capacidade do modelo preditivo aumenta e se de fato temos indicativos de variáveis que não são relevantes.

Tabela 4.3: Tabela de contingência para a Floresta Aleatória considerando 10000 árvores

	Observado		
Previsto		Perdedor	Vencedor
	Perdedor	73	15
	Vencedor	52	122

Fonte: Elaborada pelo autor

Obtemos as medidas de desempenho pela Tabela 4.3 gerada. A acurácia obtida para esse novo modelo foi de aproximadamente 74,44%, um ganho total de 0,78% se comparado com a floresta que considerou 500 árvores.

Figura 4.3: Resultados da Floresta Aleatória considerando 10000 árvores

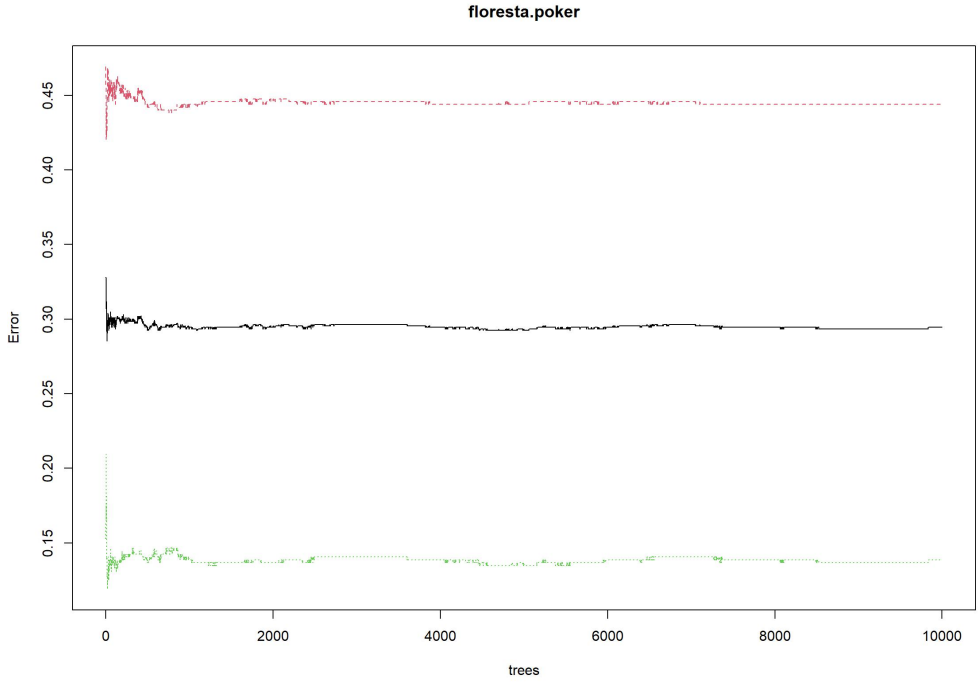
```
               Type of random forest: classification
                     Number of trees: 10000
No. of variables tried at each split: 4

        OOB estimate of  error rate: 29.45%
Confusion matrix:
         perdedor vencedor class.error
perdedor      297      237   0.4438202
vencedor       71      441   0.1386719
> importance(floresta.poker)
                   perdedor    vencedor MeanDecreaseAccuracy MeanDecreaseGini
VPIP              41.462429 117.0438351          127.3993880        44.510838
PFR                2.004124  64.8887302           65.8819916        11.380316
X3Bet             17.638869  17.5563233           31.6550565         4.141851
PostflopAgg       24.814660  29.0221607           38.6317362         7.287364
WTSD              45.134316  10.7203109           47.4626312         7.915411
FlopCBet         -17.652776  17.0492771           -0.2014167         3.224114
TurnCBet         -18.656019  32.4404762           21.8449395         6.994387
Foldvs.FlopCBet    4.744722  11.5173707           14.3879246         4.227517
Raisevs.FlopCBet   9.767361   0.4519009            8.2378163         2.569485
Squeeze           -6.106754  38.2394510           37.2620096         6.107757
Foldto3Bet        29.950441  62.7219676           75.1317282        28.825691
Callvs.3Bet       38.933789  74.9185608           96.9165836        35.824274
Raisevs.3Bet     -28.548928  51.3533388           42.1898506         7.453524
```

Fonte: Elaborada pelo autor

Observamos pela Figura 4.3 que a estimativa do erro para os OOB teve uma redução de 0,38% e o preditor FlopCBet ainda foi considerado como responsável para esse aumento do erro. Podemos notar que apesar dessa variável auxiliar no modelo preditivo para a classe vencedor, ela apresentou uma perda mais significativa ao tentar prever a classe perdedor, portanto iremos removê-la do modelo.

Figura 4.4: Relação da taxa de erro OOB e quantidade de árvores



Fonte: Elaborada pelo autor

Na Figura 4.4, a linha vermelha representa a estimativa do erro para a classe perdedora, a linha preta representa o erro médio OOB e a linha verde a estimativa de erro para a classse vencedora. Pela interpretação gráfica verificamos que a partir de aproximadamente 3000 árvores a taxa de erro médio para os OOB tende a convergir. Portanto, optamos por construir uma nova floresta considerando 3000 árvores para evitar o custo computacional desnecessário. Como removemos a variável FlopCbet, restamos com 12 variáveis preditoras e o valor do parâmetro mtry foi alterado para 3.

Figura 4.5: Resultados da Floresta Aleatória considerando 3000 árvores

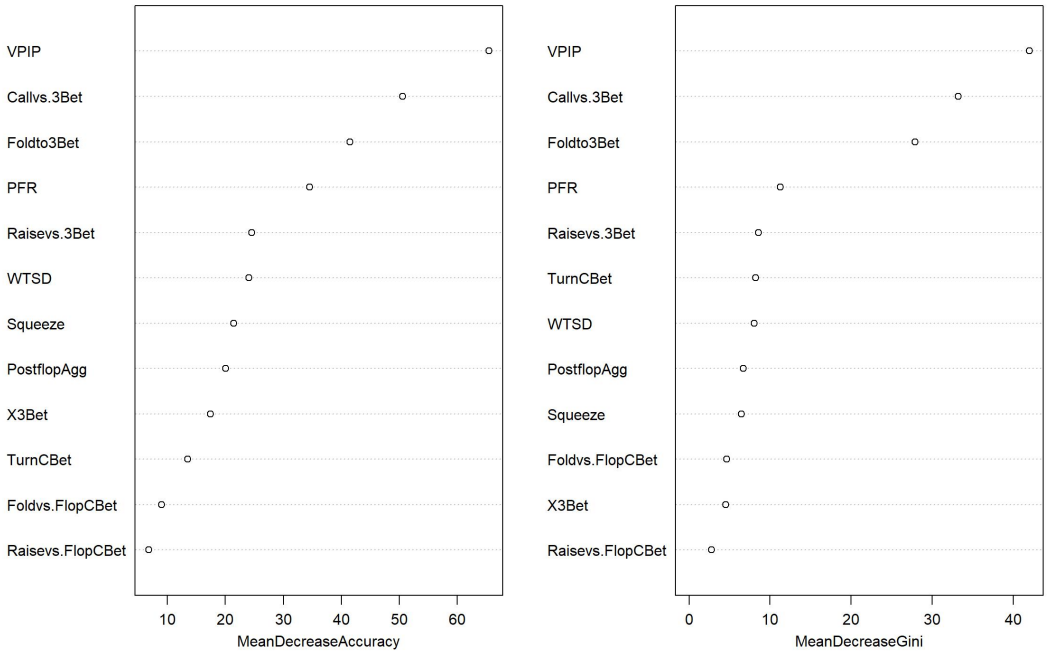
```
               Type of random forest: classification
                     Number of trees: 3000
No. of variables tried at each split: 3

        OOB estimate of  error rate: 29.25%
Confusion matrix:
          perdedor vencedor class.error
perdedor      296      238   0.4456929
vencedor       68      444   0.1328125
> importance(floresta.poker)
                  perdedor   vencedor MeanDecreaseAccuracy MeanDecreaseGini
VPIP             22.159375 59.8555933            65.495855        42.035459
PFR               1.631113 33.9545624            34.530263        11.270403
X3Bet             8.294258 11.8974879            17.438030         4.503926
PostflopAgg      14.844954 13.5867458            20.080932         6.710264
WTSD             21.953570  6.1535596            24.079815         8.069735
TurnCBet        -12.346267 19.8908720            13.519656         8.245863
Foldvs.FlopCBet   2.388738  8.0598884             8.970721         4.643777
Raisevs.FlopCBet  7.445000  0.8233143             6.762090         2.788770
Squeeze          -2.927803 22.0322332            21.461834         6.482546
Foldto3Bet       16.487444 36.0441142            41.502253        27.892983
Callvs.3Bet      20.261204 40.2662561            50.534343        33.204903
Raisevs.3Bet    -16.144213 30.1843202            24.538593         8.570444
```

Fonte: Elaborada pelo autor

Verifica-se na Figura 4.5 que tivemos uma redução de 0,20% para a estimativa de erro dos OOB e que todas os preditores estão contribuindo para a capacidade preditiva do modelo. Podemos visualizar pelos gráficos apresentados na Figura 4.6 que as variáveis VPIP, Callvs.3Bet e Foldvs.3Bet têm um nível de contribuição maior, enquanto Raisevs.FlopCbet, Foldvs.FlopCbet e TurnCBet contribuem menos.

Figura 4.6: Importância das variáveis para o modelo de classificação



Fonte: Elaborada pelo autor

Utilizamos o aprendizado do modelo final para prever a classe dos jogadores reservados à base de teste e novamente tivemos um ganho na capacidade preditiva. As medidas de desempenho podem ser extraídas da matriz de confusão apresentada na Tabela 4.4.

Tabela 4.4: Tabela de contingência para a Floresta Aleatória considerando 3000 árvores

	Observado		
Previsto		Perdedor	Vencedor
	Perdedor	73	14
	Vencedor	52	123

Fonte: Elaborada pelo autor

A acurácia total observada para esse modelo foi de aproximadamente 74,81%. Tivemos 58,4% dos jogadores observados como perdedores que foram classificados corretamente como perdedores (sensibilidade) e uma taxa de acerto de 89,78% para classificar jogadores vencedores (especificidade).

5. Conclusões

Utilizando métodos estatísticos e de aprendizado de máquina, conseguimos resultados satisfatórios para selecionar variáveis relevantes do conjunto de dados, modelar as ações dos jogadores e discriminá-los nas categorias propostas. Através da interpretação dos modelos criados pelo algoritmos de Árvore de decisão e Floresta Aleatória, foi possível perceber regras de associação coerentes para entender o perfil dos jogadores. O modelo final de classificação utilizado obteve uma acurácia de aproximadamente 0,75, sendo que na base reservada para teste, a capacidade de prever jogadores perdedores foi de 0,58 e de prever jogadores vencedores de aproximadamente 0,9.

Identificamos que as ações estratégicas são importantes para o resultado do jogo, ou seja, os jogadores que tendem a ser ganhadores se comportam de um modo diferente dos perdedores no que se diz respeito a frequência de realizar cada movimento no jogo. Inferimos que os jogadores que se envolvem em menos partidas mas se comportam de modo mais agressivo, tendo maior predominância de ações em que efetuam ou aumentam uma aposta, têm maior chance de vencer se comparados aos jogadores que participam de muitas mãos adotando uma estratégia passiva. Para estudos posteriores podem ser aplicadas técnicas de agrupamento a fim de interpretar com mais profundidade quais as características esses perfis apresentam, assim como utilizar os algoritmos de classificação para prever jogadas em etapas específicas do jogo.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Alberta, U. of: *Computer Poker Research Group*, 2018. https://poker.cs.ualberta.ca/index.html, acessado em 09/02/2022.

[2] Billings, D.: *Computer Poker*. Libratus: The Superhuman AI for No-Limit Poker, 1995.

[3] David: *Obfuscated Datamined Hand Histories*, 2011. http://web.archive.org/web/20110205042259/http://www.outflopped.com/questions/286/obfuscated-datamined-hand-histories, acessado em 15/03/2022.

[4] Desmond, K.: *Poker Theory and Analytics*, 2015. https://ocw.mit.edu/courses/sloan-school-of-management/15-s50-poker-theory-and-analytics-january-iap-2015/, acessado em 09/02/2022.

[5] Gareth James, Daniela Witten, T. H. R. T.: *An Introduction to Statistical Learning with Applications in R*. Springer, 2ª ed., 2014.

[6] HM: *Software Holdem Manager - The art + sciente of winning poker*, 2022. https://www.holdemmanager.com/hm3/, acessado em 17/03/2022.

[7] Ian H. Witten, E. F.: *Data mining practial machine learning tools and techniques*. Morgan Kaufmann Publishers, 2ª ed., 2005.

[8] Johanson, M., Bard, N., Lanctot, M., Gibson, R. e Bowling, M.: *Efficient Nash Equilibrium Approximation through Monte Carlo Counterfactual Regret Minimization*. Em *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '12, p. 837–846, Richland, SC, 2012. International Foundation for Autonomous Agents and Multiagent Systems, ISBN 0981738125.

[9] Koehrsen, W.: *Random Forest Simple Explanation*, 2017. https://williamkoehrsen.medium.com/random-forest-simple-explanation-377895a60d2d, acessado em 17/03/2022.

[10] Noam Brown, T. S.: *Superhuman AI for heads-up no-limit poker: Libratus beats top professionals*. Science, 359(6374):418–424, 2018. http://dx.doi.org/10.1126/science.aao1733.

[11] Paco Hope, S. M.: *Statistical Analysis of Texas Hold'Em*, 2009. https://law.siu.edu/_common/documents/law-journal/articles-2012/chumbley.pdf, acessado em 15/03/2022.

[12] Pang-Ning Tan, Michael Steinbach, V. K.: *Introduction to Data Mining*. Pearson, 1ª ed., 2014.

[13] Prado, S.: *Poker é reconhecido oficialmente como esporte da mente*, 2010. https://www.espn.com.br/blogs/sergioprado/118454_poker-e-reconhecido-oficialmente-como-esporte-da-mente, acessado em 09/02/2022.

[14] Quang Hung Nguyen, Hai-Bang Ly, L. S. H. N. A. A. H. V. L. V. Q. T. I. P. B. T. P.: *Influence of Data Splitting on Performance of Machine Learning Models in Prediction of Shear Strength of Soil*. Mathematical Problems in Engineering, 2021(4832864):15, 2021. https://doi.org/10.1155/2021/4832864.

[15] R: *The R Project for Statistical Computing*, 2022. http://www.R-project.org/.

[16] Records, G.: *Most players at an internet poker room*, 2009. https://www.guinnessworldrecords.com/world-records/most-players-at-an-internet-poker-room, acessado em 09/02/2022.

[17] Records, G.: *Largest prize pool for an online poker tournament*, 2020. https://www.guinnessworldrecords.com/world-records/632409-largest-prize-pool-for-an-online-poker-tournament, acessado em 09/02/2022.

[18] Sylvestrin, G.: *Como Utilizar Bootstrap Aggregation Para Classificação*, 2021. https://medium.com/datarisk-io/como-utilizar-bootstrap-aggregation-para-classificaç˜ao-37513676edfa,note={acessadoem17/03/2022},.

[19] tech, D.: *Como funciona o algoritmo árvore de decisão*, 2017. https://didatica.tech/como-funciona-o-algoritmo-arvore-de-decisao/, acessado em 17/07/2022.

[20] Teófilo, L. F. G.: *Building a Poker Playing Agent based on Game Logs using Supervised Learning*. Dissertação de Mestrado, 2010.

[21] UNICAMP: *Pôquer: Estratégias, Decisões e Negócios*, 2022. https://www.fca.unicamp.br/portal/pt-br/extensao/ext-cursos/ext-curso-extensao/641-fca-0025-poquer-estrategias-decisoes-e-negocios.html, acessado em 09/02/2022.

[22] Velho, N.: *IMPLEMENTAÇÃO ITERATIVA DO ALGORITMO COUNTERFACTUAL REGRET MINIMIZATION*, 2014. https://paginas.fe.up.pt/˜ee09041/ContextoMotivacao.html, acessado em 03/03/2022.

[23] Wikipedia: *Libratus*, 2017. https://en.wikipedia.org/wiki/Libratus, acessado em 09/02/2022.

GLOSSÁRIO

- **Hold'em:** Modalidade do pôquer em que cada jogador recebe duas cartas.
- **Pote:** O pote é o valor total de fichas acumuladas na mesa do jogo em uma única partida.
- **Blinds:** Apostas mínimas de uma determinada mesa do jogo.
- **Stack:** Quantidade de fichas que cada jogador tem disponível na mesa em que está participando.
- **Bet:** Ação de realizar uma aposta.
- **Call:** Ação de pagar uma aposta.
- **Check:** Ação de passar a vez.
- **Raise:** Ação de aumentar a uma aposta efetuada previamente por outro jogador.
- **Fold:** Ação de desistir ao enfrentar uma aposta.
- **Pré-Flop:** Etapa inicial do pôquer onde os jogadores recebem suas cartas e ocorre a primeira rodada de apostas.
- **Flop:** O termo flop representa a rodada em que três cartas comunitárias são expostas aos jogadores.
- **Turn:** A quarta carta comunitária exposta.
- **River:** A quinta e última carta comunitária exposta.
- **Showdown:** Momento que os jogadores envolvidos revelam suas cartas para comparar quem tem a melhor combinação de jogo.
- **PostFlop:** Representa o momento de acontecimentos após a abertura do Flop.
- **Hero:** O nome dado ao jogador que relatou sua partida para o(s) espectador(es) contemplar(em).
- **VPIP:** Estatística que revela a frequência em que o jogador optou por participar após receber suas cartas na etapa inicial do jogo.

Código R

A seguir, apresentamos o código R utilizado na metodologia do trabalho.

```
# Iniciando os pacotes e bibliotecas necessarias.
library(rpart)
library(rpart.plot)
library(randomForest)

# Entrando com os dados:
# Base contendo todos os jogadores. Mesa com 4-6 pessoas. HoldemNL.
X <- read.csv2("Base4max.txt",sep="",h=T)
str(X)
summary(X) # detectando anomalias

# Removendo as linhas que contem n.as:
indices <- c()
for (j in 1:nrow(X)) {
  if(any(is.na(X[j,]))){
    indices <- c(indices, j)
  }
}
X <- X[-indices,]

# Considerando 14 variaveis removemos 130307 linhas com n.as

# Transformando a classe na categoria de variavel qualitativa:
Classe <- c()
for (j in 1:46112){
  if(X$NetbbWon[j] > 700){
    Classe[j] <- "vencedor"
  }else if (X$NetbbWon[j] < -650){
    Classe[j] <- "perdedor"
  } else {
    Classe[j] <-"empatado"
```

```
  }
}
X$Classe <- Classe
mean(Classe=="vencedor")
mean(Classe=="perdedor")

# Removendo a classe empatado
indices1 <- c()
for (j in 1:46112){
  if(X$Classe[j] == "empatado"){
    indices1 <- c(indices1, j)
  }
}
X <- X[-indices1,]

# Adicionando a variavel classe na base de dados.
X$Classe <- as.factor(X$Classe)
# Removendo a variavel net.won:
X <- X[,-2]
str(X)
# Selecao de atributos.
# Variaveis removidas por nao representarem frequencias de acoes.
X <- X[,-1]
X <- X[,-5]
summary(X$Classe)

# Separando a base em treino 80% e teste 20%:
set.seed(8)
df <- X[sample(nrow(X)),]
treino <- df[1:1046,]
teste <- df[1047:1308,]

# Criando modelo da arvore de decisao:
arvore.poker2 <- rpart(formula = Classe~., data = treino,
method="class", control=rpart.control(xval=10,maxdepth=6))
rpart.plot(arvore.poker2, extra=101)
predicao2 <- predict(arvore.poker2, newdata=teste, type ="class")
mean(teste$Classe == predicao2) # Acuracia 71%.
table(predicao2, teste$Classe) # Matriz de confusao
```

```
# Criando modelo da floresta aleatoria:
floresta.poker <- randomForest(formula= Classe ~ ., data=treino,
ntree=500, maxnodes=15,mtry=4,importance=TRUE)
predicted= predict(floresta.poker, newdata=teste)
mean(teste$Classe == predicted,type="class") # Acuracia 73.66%.
table(predicted, teste$Classe) # Matriz de confusao
floresta.poker
importance(floresta.poker)

# Criando modelo da floresta aleatoria para 10000 arvores.
floresta.poker <- randomForest(formula= Classe ~ ., data=treino,
ntree=10000, maxnodes=15,mtry=4,importance=TRUE)
predicted= predict(floresta.poker, newdata=teste)
mean(teste$Classe == predicted,type="class") # Acuracia 74,42%.
table(predicted, teste$Classe) # Matriz de confusao
floresta.poker
importance(floresta.poker)
# Relacao grafica taxa de erro e quantidade de arvores.
plot(floresta.poker)

# Removendo a variavel FoldCbet.
treino <- treino[,-6]
teste <- teste[,-6]

# Criando modelo da floresta aleatoria para 3000 arvores:
floresta.poker <- randomForest(formula= Classe ~ ., data=treino,
ntree=3000, maxnodes=15,mtry=3,importance=TRUE)
predicted = predict(floresta.poker, newdata=teste)
mean(teste$Classe == predicted,type="class") # Acuracia 74,8%.
table(predicted, teste$Classe) # Matriz de confusao.
floresta.poker
importance(floresta.poker)
# Representacao grafica da importancia das variaveis.
varImpPlot(floresta.poker)
```