UNIVERSIDADE DO SAGRADO CORAÇÃO

GIOVANE MORBI

TREINAMENTO DE UM JOGADOR AUTÔNOMO DE PÔQUER UTILIZANDO APRENDIZAGEM DE MÁQUINA

BAURU

2017

GIOVANE MORBI

TREINAMENTO DE UM JOGADOR AUTÔNOMO DE PÔQUER UTILIZANDO APRENDIZAGEM DE MÁQUINA

Trabalho de Conclusão de Curso apresentado ao Centro de Ciências Exatas e Sociais Aplicadas como parte dos requisitos para a obtenção do título de bacharel em Ciência da Computação, sob orientação do Prof. Dr. Elvio Gilberto da Silva.

BAURU
2017

Morbi, Giovane

M832t

Treinamento de um jogador autônomo de pôquer utilizando aprendizagem de máquina / Giovane Morbi. -- 2017.

52f. : il.

Orientador: Prof. Dr. Élvio Gilberto da Silva.

Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) - Universidade do Sagrado Coração - Bauru - SP

1. Aprendizagem de máquina. 2. Ensemble learning. 3. Machine learning. 4. Pôquer. 5. Gradient Boosting Regressor. I. Silva, Élvio Gilberto da. II. Título.

GIOVANE MORBI

TREINAMENTO DE UM JOGADOR AUTÔNOMO DE PÔQUER UTILIZANDO APRENDIZAGEM DE MÁQUINA

Trabalho de Conclusão de Curso apresentado ao Centro de Ciências Exatas e Sociais Aplicadas como parte dos requisitos para a obtenção do título de bacharel em Ciência da Computação, sob orientação do Prof. Dr. Elvio Gilberto da Silva.

Bauru, 29 de Novembro de 2017.

Banca examinadora:

_____
Prof. Dr. Elvio Gilberto da Silva
Universidade do Sagrado Coração

_____
Prof. Me. Henrique Pachioni Martins
Universidade do Sagrado Coração

_____
Prof. Me. Patrick Pedreira Silva
Universidade do Sagrado Coração

SUMÁRIO

RESUMO

O pôquer jogo tradicional dos cassinos, vem ganhando cada vez mais adeptos, pessoas que estão deixando suas profissões para tornarem-se jogadores profissionais de pôquer, visto que o mercado para a prática desse esporte tem sido bem atrativa, possibilitando participar de campeonatos mundiais e gerando alta rentabilidade, pois as apostas têm sido cada vez mais elevadas, o jogo passou de apenas prática de lazer e jogos clandestinos, para esporte, que está sendo praticado por todo o mundo, tendo diversos campeonatos realizados durante o ano, descobrindo novos competidores e as apostas nessas mesas sempre girando em altos valores.

Com a procura do pôquer como fonte de renda para muitos, e a competição tornando-se cada vez mais acirrada, apenas saber jogar o jogo não é suficiente, e os jogadores que levam o assunto cada vez mais a sério, buscam treinar e aprimorarem-se para conquistar a vitória. Durante seus aprimoramentos, eles buscam diversas áreas para que possam auxiliar aos seus jogos. Área como a matemática, com os cálculos de probabilidade, área da psicanálise, para estudar o perfil de seus oponentes, através do comportamento deles durante os jogos, e também para conseguirem disfarçar atitudes durantes os jogos, para seus adversários não perceberem.

Toda essa procura de aprimoramento e desenvolvimento de jogadas, não é diferente que busquem apoio à área de tecnologia, e este trabalho visou viabilizar e evidenciar a capacidade de implementar a área de aprendizagem de máquina a jogos como o pôquer, demonstrando que é possível desenvolver e treinar um jogador autônomo, capaz de jogar pôquer e conquistar vitórias. Através de demonstrações narrativas de jogadas, simulando uma mesa de pôquer, com demais oponentes, junto ao treinamento do algoritmo de ensemble learning, apoiado à uma biblioteca do MIT.

Palavras-chaves: Pôquer, aprendizagem de máquina, ensemble learning, gradient boosting regressor.

ABSTRACT

Traditional casino gambling has been gaining more and more supporters, people who are leaving their professions to become professional poker players, since the market for the practice of this sport has been very attractive, allowing to participate in world championships and generating high stakes, since gambling has been increasingly high, the game has gone from recreation and clandestine sports, which is being practiced all over the world, with several championships held during the year, discovering new competitors and betting on these tables always turning in high values.

With the pursuit of poker as a source of income for many, and the competition becoming more and more fierce, just knowing how to play the game is not enough, and players who take the matter more and more seriously seek to train and improve, to win the victory. During their upgrades, they seek out various areas to help their games. Area such as mathematics, probability calculations, area of psychoanalysis, to study the profile of their opponents, through their behavior during games, and also to disguise attitudes during games, for their opponents do not realize.

All this demand for improvement and development of plays is no different than seeking support in the area of technology, and this work aimed to make feasible and demonstrate the ability to implement the area of machine learning to games such as poker, demonstrating that it is possible to develop and to train an autonomous player, able to play poker and win victories. Through narrative demonstrations of plays, simulating a poker table, with other opponents, along with the training of the ensemble learning algorithm, supported by an MIT library.

Keywords: Poker, machine learning, ensemble learning, gradient boosting regressor.

1 INTRODUÇÃO

O tradicional pôquer, jogado nos cassinos pelo mundo a fora, com diversos jogadores em suas mesas, realizando altas apostas e gerando recompensas inimagináveis, vem destacando-se de demais jogos de cartas, passando a ser considerado um esporte, ganhando cada dia mais adeptos ao esporte e virando um ícone mundial, tendo competições mundiais com competidores espalhados pelo mundo e realizando jogos on-line em tempo real.

Torneios que se iniciam em mesas on-line e são finalizados em mesas de cassinos famosos nos estados unidos, movendo milhares de dólares em apostas e premiações aos ganhadores. Cada vez mais, pessoas que tinha suas atividades rotineiras, como ser médico, cientista, professor estão associando à sua carreira e rotina o pôquer, muitos deixam de exercer sua profissão para tornarem-se jogadores profissionais de pôquer.

Todos buscam aprimorarem suas técnicas, suas jogadas, levando ao aprendizado da matemática aplicada à probabilidade e estatística, associando também à programação neurolinguística, para estudar e aprender o perfil e atitude de seus oponentes.

E como a demanda vem crescendo, não é diferente que eles iriam deixar de recorrer as áreas da computação, buscando algoritmos avançados e formas de facilitar suas jogadas e até conseguirem prever possíveis jogadas dos adversários e o que virá a ser utilizado por ele. É possível também levantar um histórico de cada jogo, e associar tudo isso à inteligência artificial, para gerar relatórios, históricos de jogos e como cada jogador se comporta, associando tudo isso ao aprendizado de máquina, podemos ter um jogador autônomo capaz de realizar jogadas como um ser humano.

Este trabalho consiste em apresentar um jogador autônomo capaz de efetuar jogadas no modo "*Texas Hold'em no limit",* utilizando aprendizagem de máquina, apoiado à uma biblioteca modelo do MIT (Instituto Tecnológico de Massachusetts), utilizando um algoritmo *"ensemble".*

2 OBJETIVOS

A seguir serão apresentados os objetivos geral e específicos desta pesquisa.

2.2 OBJETIVO GERAL

Desenvolver um jogador autônomo de pôquer apoiado à um modelo de aprendizagem de máquina.

2.3 OBJETIVOS ESPECÍFICOS

a) Levantar regras e estratégias do pôquer.
b) Buscar as melhores jogadas com a maior chance de vitória.
c) Estudar o algoritmo mais eficaz para solucionar o problema proposto.
d) Implementar o algoritmo escolhido relacionado ao aprendizado de máquina.
e) Elaborar um agente autônomo.
f) Demonstrar resultados obtidos com o aprendizado de máquina utilizada no pôquer.
g) Apresentar os resultados obtidos com o jogador autônomo e o algoritmo aplicado.

3 REVISÃO DA LITERATURA

Neste tópico é apresentado a revisão bibliográfica acerca do projeto proposto.

3.2 REGRAS DO POKER TEXAS HOLD'EM

O pôquer é encontrado em diversas civilizações através do mundo, como por exemplo, Pérsia, Índia, Alemanha e França, essas civilizações podem por sua vez ter influenciado como o jogo é hoje, o pôquer conhecido nos dias de hoje, teve origem no final do século XIX, em Luisiana, terra colonizada por Franceses. (FERREIRA; JEFFERSON, 2017).

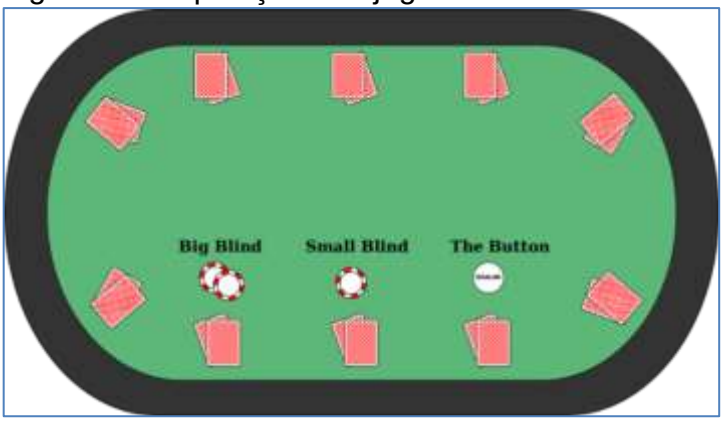
As regras do pôquer são bem simples, a mais relevante de todas é referente às combinações das cartas. O objetivo principal é vencer, jogo que testa as suas estratégias e capacidades. (DICAS DE POKER, 2008).

O jogo é formado por 52 cartas de baralho, o mínimo de jogadores para uma partida são dois e o máximo dez pessoas em uma mesma mesa, os jogadores são distribuídos em uma mesa oval, para que não haja a possibilidade de nenhum ver a carta de seu oponente, tendo como sentido a distribuição de caras e ações o sentido horário.

3.3 PRIMEIRA RODADA (PRÉ-FLOP)

Os jogadores ficam dispostos na mesa, um ao lado do outro, formando praticamente uma elipse na mesa, alguns jogadores durante a partida possuem uma posição privilegiada, posições denominadas de Dealer (The button), Small Blind (Menor aposta cega), Big Blind (Maior aposta cega), essas posições rodam a mesa, sendo assim todos os jogadores da mesa tem a chance de ser o Dealer, Small Blind ou Big Blind. Na Figura 1 é possível notar como é feita distribuição em uma mesa de pôquer.

Figura 1 - Disposição dos jogadores na mesa com os botões



Fonte: Site do Instructables[1]

Essas posições, que rodam por todos os jogadores, são as apostas cegas, dadas no início de cada rodada, o jogo de pôquer possui uma aposta obrigatória, pelo qual os competidores se enfrentam, salvo o pôquer por lazer sem apostas. As apostas servem para que o jogo tenha um início e um final, senão o jogo seria interminável, as posições privilegiadas andam em sentido horário, onde o *Small Blind torna-se Dealer (The Button), Big Blind passa a ser Small Blind* e o próximo à esquerda do *Big Blind* será o novo *Big Blind.* presente na mesa, e a rodada inicia-se perguntando ao jogador ao lado do B*ig Blind* se ele tem interesse em participar da

[1] http://www.instructables.com/id/How-To-Play-Texas-Holdem/step2/Blinds/

rodada, casa ele aceite, poderá pagar a aposta feita pelo *Big Blind* ou cobrir sua aposta, realizando assim o aumento do valor e esse novo valor é que será cobrado ao próximo participante, os passos se repetem até chegar no *Small Blind,* mas como ele já havia pago metade da aposta antes das cartas serem dadas, ele tem a opção de igualar a aposta para continuar na próxima jogada ou desistir. O último a ser questionado ainda nessa rodada é o *Big Blind*, ele poderá desistir quanto aumentar as apostas ou apenas pedir que prossigam com o jogo, caso ele decida aumentar, os passos anteriores serão retomados, para que todos os participantes igualem suas apostas ao valor atual, caso ele peça mesa, a próxima rodada será iniciada.

3.4 SEGUNDA RODADA (FLOP)

A rodada chamada de FLOP, inicia-se depois que houve a equalização de todas as apostas da primeira rodada, finalizando assim no *Big Blind.*

Na segunda rodada é aberto três cartas à mesa, elas que serão utilizadas para os jogadores realizarem suas combinações com as suas duas cartas recebidas no início do jogo.

Com isso reinicia a regra das apostas, podendo ser aumentadas, haver desistências ou pedidos de mesa, conforme descrito no item anterior, seguindo a regra que finaliza e dá-se início a próxima rodada sempre no *Big Blind.*

3.5 TERCEIRA RODADA (TURN)

A terceira rodada é iniciada abrindo apenas uma carta à mesa, assim os jogadores já possuem quatro cartas em suas combinações, e o *Big Blind* dará início à uma nova seção de apostas, terminando as apostas no jogador anterior a ele, permanecendo as mesmas regras de apostas, caso um jogador aumente todas terão que ser igualadas.

3.6 QUARTA RODADA (RIVER)

Por fim chegada a quarta rodada, onde a última carta é aberta na mesa, sendo assim, tem-se três cartas à mesa e duas na mão de cada jogador e cada jogador faz uma combinação de cinco cartas, esta é a última rodada, onde serão feitas as últimas apostas ou desistências, a rodada só termina quando o último jogador igualar a maior aposta.

3.7 FIM DE JOGO (SHOWDOWN)

No *Showdown* ficam apenas os jogadores que não desistiram nas etapas anteriores, nela cada jogador irá abrir suas cartas que estavam consigo, para decidir quem realizou a melhor combinação e irá levar o pote daquela partida, o pote é toda a aposta acumulada no ciclo, contendo o valor das apostas iniciais e das subsequentes.

O *showdown* é inciado pelo jogador depois do *Dealer (The Button)*, chamado de intermediário, todos os jogadores que permaneceram irão mostrar em sequência da mesa, suas cartas para verificar quem fez a combinação de maior valor, a sequência é repetida para que seja feita um cômputo do vencedor.

Nas regras não existe obrigatoriedade do segundo até o último jogador revelar suas cartas, caso ele opte em não revelar, é declarado perdedor automaticamente.

3.8 RANKING

A decisão na última rodada, dá-se pela melhor combinação de cinco cartas, estabelecido nas regras do jogo. Ganha quem tiver a melhor mão (combinação de cartas).

A ordem que define a maior da menor carta, sequencialmente é: A, K, Q, J, 10, 9, 8, 7, 6, 5, 4, 3, 2.

O ranking das mãos de pôquer segundo “pokerstars”:

a) **Straight Flush:** Cinco cartas em ordem numérica, todas do mesmo naipe. No caso de um empate: A carta de maior valor no topo da sequência vence. O melhor Straight Flush possível é conhecido como Royal Flush, que consiste na sequência

de Ás, Rei, Dama, Valete e Dez do mesmo naipe. O Royal Flush é uma mão imbatível.

b) **Quadra:** Quatro cartas do mesmo valor, e uma outra carta como ‘Kicker’. No caso de um empate: A maior quadra vence. Em jogos com cartas comunitárias onde os jogadores podem conseguir a mesma quadra, a maior quinta carta (Kicker) vence.

c) **Full House:** Três cartas do mesmo valor, e duas de outras cartas diferentes de mesmo valor. No caso de empate: As três maiores cartas do mesmo valor vencem o pote. Em jogos com cartas comunitárias os jogadores podem conseguir as mesmas três cartas de valor igual, vence aquele com o par de maior valor.

d) **Flush:** Cinco cartas do mesmo naipe. No caso de um empate: o jogador com a carta de maior valor vence. Se necessário, a segunda carta, a terceira carta a quarta carta e a quinta carta podem ser utilizadas para desempate. Se as cinco cartas forem do mesmo valor, o pote é dividido. O naipe nunca é utilizado como critério de desempate no pôquer.

e) **Sequência:** Cinco cartas em sequência. No caso de empate: a carta de maior valor no topo da sequência vence. Observação: o Ás pode ser utilizado no topo ou na parte baixa da sequência, e é a única carta que pode ser utilizada dessa forma. A, K, Q, J, T é a maior sequência (Ás alto ou Ace High). 5, 4, 3, 2, A é a menor sequência (5 alto ou Five High).

f) **Trinca:** Três cartas do mesmo valor, e duas outras cartas não relacionadas. No caso de empate: A trinca de maior valor vence. Em jogos com cartas comunitárias onde os jogadores podem ter a mesma trinca, ganha o jogador com a maior carta além das três de mesmo valor, e se necessário a segunda maior carta será utilizada para desempate.

g) **Dois Pares:** Duas cartas de um mesmo valor, outras duas cartas diferentes de mesmo valor, e uma outra carta não relacionada. No caso de empate: O par maior vence, se os jogadores possuírem o mesmo par mais alto, o segundo par decide o vencedor. Se os dois jogadores tiverem pares idênticos, a quinta carta define o vencedor.

h) **Um Par:** Duas cartas de mesmo valor, e três outras cartas não relacionadas. No caso de empate: O par maior vence. Se dois jogadores tiverem o mesmo par, a maior carta fora do par define o vencedor, e se necessário a segunda maior carta e

a terceira maior carta pode ser utilizada para desempate. **Carta Alta:** Qualquer mão que não esteja nas categorias acima.

i) No caso de empate: A maior carta vence, e se necessário, a segunda carta, a terceira carta, a quarta carta e a quinta podem ser utilizadas para desempate.

3.9 ANÁLISE DO PÔQUER

Não existe a receita mágica ou regras básicas a serem seguidas que irá te tornar vencedor campeão no pôquer, mas conteúdo o suficiente para apoiar à decisão correta, desconsiderando possíveis trapaças no jogo. Extraindo informações de cada jogada e rodada, como posição de cada jogador, valor do pote, número de apostas, desistências, cartas abertas na mesa e na mão do jogador, tem de ação e reação de cada jogador presente na mesa, torna possível elaborar a melhor estratégia para jogar.

Todos esses fatores podem ser combinados no algoritmo de aprendizado de máquina, fazendo com que se haja um aprendizado em cada jogada e a cada partida, permitindo que o computador realize os cálculo complexos que levaríamos muito tempo para realizarmos para tentar descobrir qual seria a possível carta à vir ser aberta na mesa ou se algum jogador tem a pretensão de aumentar ou desistir das apostas, a única variável que um algoritmo de aprendizado de máquina não é capaz ainda de captar durante um jogo de pôquer está ligado à emoção do outro jogador, item este, crucial para uma partida de pôquer.

Visto que muitos jogadores analisam todas as ações de seus adversários, vem no seu olho se estão dispersos, disfarçando, em suas atitudes com as mãos, jogadores que conseguem fazer uma leitura profunda de seu adversário possui vantagem sobre ele, tendo a chance de levar o pote consigo. Porém em algumas rodadas que o ser humano acaba desistindo por agir através do impulso, o jogador autônomo não iria desistir pela quantidade de informações que ele coletou e o cálculo de probabilidade se ser uma boa jogada para a mão dele.

3.10 CÁLCULO DE OUTS

A matemática está presente na maioria dos jogos de cartas, ou na probabilidade da próxima carta ser sacada do baralho ou na combinação feita que irá somar os pontos, no pôquer não é diferente ela está presente.

Em resumo o cálculo de Outs é a quantidade de cartas que faltam para formar uma mão. Exemplo: se o jogador possui A e J do mesmo naipe, e na mesa o Flop já saiu, formado por duas cartas também do mesmo naipe de sua mão, independentemente do valor, e uma outra de naipe qualquer, então o jogador está esperando um Flush. Para calcular o Outs é simples, considera-se para isso a quantidade total de cartas para formar a mão menos o que já está na rodada. Ainda falta sair para o jogador treze cartas do naipe que está em sua mão, mas como já saiu quatro, então ainda restam nove cartas, assim ele possui nove Outs.

3.11 CÁLCULO DE ODDS

De acordo com Mojave(2008): “Basicamente para saber se vale a pena ou não pagar uma aposta, ou mesmo no tamanho de uma aposta que iremos aplicar.”

Odds é a razão entre a probabilidade de vencer e perder. Exemplo: se um jogador está numa mesa que já saiu o Flop, nela ele terá as A e 8 na mão e na mesa tem K, 8, 3 e 2.

O jogador terá um par médio de 8. Deve-se então calcular o númer de Outs antes e depois realizar o cálculo de Odds em relação à jogada. O número de Outs então é de três Ás e dois outros, somando os dois tem-se cinco Outs.

Como foi dito anteriormente, é a razão entre ganhar e perde, para descobrir esta razão de ganho são utilizadas duas condições:

I. Para mensurar se há melhoria da mão depois do Flop, ou seja no Turn/River, a probabilidade de ganho é:

$$[PROBABILIDADE\ DE\ GANHO] = [NÚMERO\ DE\ OUTS]\ x\ 4 \qquad (1)$$

II. Para medir se há melhoria da mão depois do Flop, ou seja no River, a probabilidade de ganho é:

$$[PROBABILIDADE\ DE\ GANHO] = [NÚMERO\ DE\ OUTS]\ x\ 2 \qquad (2)$$

A probabilidade de perda é calculada numa equação simples que é cem por cento menos a probabilidade vista anteriormente:

$$[PROBABILIDADE\ DE\ PERDA] = 100\% - [PROBABILIDADE\ DE\ GANHO] \qquad (3)$$

3.12 POT ODDS

Crucial para decisões em relação ao pote da mesa, é a variável calculada para obter mais precisão nas decisões a serem tomadas.

Conforme MOJAVE(2016): “Pôquer envolve uma imensa gama de elementos que devem ser levados em consideração, e os jogadores devem estar aptos a fazerem uso de todos eles para sempre manterem o controle da situação. ”

O cálculo consiste entre a razão do pote e a Odds da mão do jogador, se a razão do pote for maior que a mão do jogador, o jogador deve pagar.

Por exemplo, se há um FLUSH DRAW de 9 Outs e o pote é de R$10 e um oponente aposta R$2, somando-se ao pote, tem-se R$12. A relação então de Odds do pote é de 10:2 e a da mão é 4:1. Se a razão do POT ODDS é amior que a razão da mão, então o jogador deve pagar.

Em uma outra simulação, levando em consideração a desistência da mão, supondo que o um jogador esteja esperando uma carta para realizar um Straight, e que então há 4 Outs na jogada, Calcula-se o Odds da mão, tem-se 6:1. Se o oponente aposta R$10 em um pote que já continha R$40, então a razão é de 40:10 ou 4:1. Como nesse caso, a razão da mão em relação ao pote é maior, então como boa prática, deve-se desistir.

4 INTELIGÊNCIA ARTIFICIAL

Existem registros evidenciando no século XVII e XVIII a presença de mitos ou lendas sobre criaturas artificiais, o filósofo Descartes (1596-1650), argumentava em suas obras que autômatos jamais teriam habilidades mentais semelhantes aos homens, já o filósofo Mettrie era contrário a Descartes, dizia que um dia seríamos capazes de construir um homem mecânico capaz de desenvolver faculdades mentais. No século XIX, o industrialismo tornou-se o foco dos assuntos esquecendo-se das discussões dos filósofos, porém na literatura através do movimento romântico, as criaturas artificiais voltaram a ser lembradas, por exemplo o romance Frankenstein, publicado nesta época. (MATTOS, 2005).

Com a 2ª Guerra Mundial, ela trouxe o advento do avanço na tecnologia e em demais áreas, junto vinha a pressão na comunidade científica, eles precisam estar cada vez mais atentos, e desenvolverem a todo instante uma nova saída para superar seus inimigos, e com isso reaparece a inteligência artificial, onde era preciso desenvolver canhões antiaéreos para abaterem os bombardeiros, com um sistema que corrigisse problemas eventuais de mira, desvio do alvo, foram realizados experimentos em cérebros humanos, ao fim da 2ª Guerra Mundial, cientistas possuíam uma gama de dados registrados através dos experimentos, e até invenções importantes na área.

Nesta mesma época, durante a 2ª Guerra Mundial, Alan Turing, apresenta os princípios do funcionamento de um futuro computador, estabelecendo assim uma analogia entre o cérebro humano e os computadores, que antes eram pessoas que exerciam essa atividade. Isso só foi possível ser estabelecido por psicólogos, neurofisiológicos e engenheiros eletrônicos, pois eles puderam identificar que nossos neurônios se assemelhavam aos circuitos elétricos de um computador. (MATTOS, 2005).

O termo inteligência artificial originou-se em 1956, no encontro de Dartmouth, presentes no evento estavam, Allen Newell, Herbert Simon, Marvin Minsky, Oliver Selfridge e John McCarthy, a partir deles foi introduzido o processamento simbólico invés de um sistema baseado em números.

Para Bellman (1978), inteligência artificial é a propriedade de um artefato de poder resolver problemas que se fossem resolvidos por um ser vivo, ele seria considerado inteligente.

Para Winston (1992), IA é o estudo das ideias que permitem aos computadores serem inteligentes.

Conceitua-se Inteligência Artificial como um ramo de pesquisa da área de ciência da computação, onde a busca através de símbolos de natureza não numéricos, resultando em aplicações que antes necessitavam de seres humanos e suas capacidades de raciocínio e perícia, simulando assim a capacidade de um humano, capaz de pensar e resolver problemas ou seja ser inteligente.

O estudo da IA divide-se em 4 grande áreas, a primeira ligada a redes neurais, onde liga a capacidade dos computadores a reconhecerem padrões, a segunda relacionado a biologia molecular, estudando a possibilidade de construir vida artificial, a terceira que está ligado a robótica, interagindo com a biologia, procurando construir máquinas capazes de alojarem vida artificial e por fim a quarta, que é a clássico IA, inicialmente interligada a psicologia, desde os anos 70 à epistemologia, e desde os anos 80 à sociologia, tentando representar na máquina os mecanismos que os seres humanos possuem de raciocínio e procura. (Terry Winograd).

Através da evolução computacional a IA tornou-se mais forte, possibilitando um avanço na área da computação, a máquina tornou-se capaz de realizar análises e sintetizar a voz humana. Inicialmente buscava-se na IA reproduzir a capacidade de pensamento do ser humano, porém como toda pesquisa evolui, os cientistas aprofundaram suas pesquisas e buscaram aprimorar a máquina, para não só pensar como um ser humano, mas capaz de sentir e ser criativa além do auto aperfeiçoamento, filmes como o “Homem bicentenário” e “A.I. (Inteligência Artificial)”, retratam muito bem essa vontade da máquina se tornar humano, o filme “Eu, Robô” e “Homem de Ferro” também retratam a máquina capaz de se auto aperfeiçoar.

Constantemente diversos pesquisadores estudam essa área e sempre buscam o progresso até atingirem a inteligência similar ao do ser humano, apesar de ser um processo lento, todos esses progressos já existentes impactam e trazem benefícios a outras áreas, como o diagnóstico médico, jogos, planejamento e pesquisas automatizados. Como em qualquer outro campo de pesquisa, existem os apoiadores e os que estão contra as máquinas terem pensamento e resolverem os problemas corriqueiros dos seres humanos. Ainda existe uma barreira de preconceitos que dia após dia vem sendo quebrada, alguns céticos não creem que

as máquinas possam ter a mesma capacidade de pensar como nós, outros por sua vez pensam que a máquina irá substituir a sua posição em seu local de trabalho e acabam que não contribuem para o aprimoramento das máquinas.

A IA está presente para auxiliar a resolvermos problemas, através do conhecimento adquirido e manipulando-o. Para podermos resolver qualquer problema é necessário termos o mínimo de conhecimento sobre o problema e utilizar a melhor técnica de busca para a solução.

Nos métodos que abordam a solução de problemas podemos citar:

a) IAS: Inteligência artificial simbólica, baseada no princípio do sistema simbólico, onde é simulado o comportamento inteligente, considera-se a probabilidade, baseada em cálculos matemáticos de probabilidade, utilizando como por exemplo o teorema de Bayes, para realizar a inferência. (MATTOS, 2005).
b) IAC: Inteligência artificial conexionista, apoiada através de redes neurais possui os pensamentos interligados, como um neurônio e suas conexões biológicas. (MATTOS, 2005).
c) IAE: Inteligência artificial evolucionária, aplica-se inteligência ao comportamento da população estudada, onde essa muda seu comportamento para adaptar-se ao meio que se encontra.
d) IAH: Inteligência artificial híbrida, quando busca atingir vantagens que mais de uma inteligência oferece, gera-se uma IAH, para obter os melhores resultados e a solução dos problemas.

5 APRENDIZAGEM DE MÁQUINA

Para otimizar as interpretações dos jogos e jogadores, será utilizado aprendizagem de máquina, que é uma subcategoria da disciplina de inteligência artificial. Seu foco são algoritmos competentes em aprender, adaptar-se as instruções e parâmetros baseados no conjunto de dados reservados para o treinamento, otimizações que sempre visam a redução do custo do processamento das funções.

Existem 3 tipos categorias de algoritmos para a aprendizagem de máquina, aprendizado supervisionado (supervised learning), aprendizado não supervisionado (unsupervised learning) e o aprendizado de reforço (reinforcement learning).

Supervised Learning é um algoritmo que consiste em uma variável alvo ou variável resultado, que deve ser prevista a partir de um conjunto de dados pressupostos, a partir deste conjunto de variáveis geramos uma função, onde ela mapeia as entradas para as saídas desejadas, o processo de treinamento permanece até que o modelo atinja um nível de precisão nos dados de treinamento. Algoritmos exemplos para aprendizado supervisionado são: Regression, Decision Tree, Random Forest, KNN, Logistic Regression.

Unsupervised Learning é um algoritmo onde não há variável alvo ou variável resultado para prever ou estimar, ele é utilizado para agrupar uma população em diferentes grupos, utilizado para segmentar clientes em diferentes grupos para uma intervenção específica, exemplos de algoritmos para aprendizagem não supervisionada: Apriori algorithm, K-Means.

Reinforcement Learning é um algoritmo onde o treinamento é feito para uma decisão específica, a máquina fica em um treinamento constante de tentativa e erro, ela aprende com a experiência anterior, tenta capturar o melhor conhecimento da experiência anterior para assim poder tomar decisões precisas, algoritmo para aprendizado de reforço: Markov Decision Process.

A comunidade esportista vem se interessando cada vez mais por aprendizagem de máquina e o reconhecimento estatísticos de padrões, por exemplo o Google uniu-se à comunidade e desenvolveram uma IA que foi capaz de derrotar o campeão mundial de AlphaGO, jogo que possui diversas combinações possíveis de vencer.

É oferecido através da aprendizagem de máquina uma maior objetividade nas predições de jogos, além de aumentar a capacidade e assertividade no processo de tomada de decisões.

A aplicação de machine learning junto à jogos dá-se inicialmente através de redes neurais artificiais, gerando grande entusiasmo e incentivando o desenvolvimento de demais algoritmos adequados, soluções focada em demais áreas, uma das propriedades interessantes das redes neurais é a capacidade de aproximar qualquer função através do processo de aprendizagem de máquina, também chamado de aprendizado ou treinamento, onde um conjunto de parâmetros na rede neural através de uma simples conexão de dados não lineares.

Assuntos como aprendizagem de máquina veem cada vez mais amadurecendo e gerando um maior esforço no desenvolvimento de bases teóricas mais aprofundadas, ajudando assim, tornar cada vez melhor a compreensão da teoria de aproximação de algoritmos, a maior diferença entre aprendizagem de máquina e estatística, é dada que o foco da aprendizagem de máquina apoia-se como na teoria da complexidade computacional, computabilidade e a generalização, alguns aspectos são tratados como um casamento entre a matemática aplicada e ciência da computação.

Na área de machine learning, devemos nos atentar à análise e desenvolvimento de métodos lineares supervisionados e não supervisionados, de extração de dados e os padrões de classificação, métodos lineares são mais interessantes no momento de decisões estratégicas, pois são de fácil compreensão e análise, tornando a interpretação mais fácil em relação a métodos não lineares de classificação e regressão, construídos por uma rede neural artificial. No modelo linear, pode se observar uma estrutura consistente, auxiliando o processo de formação de imagens ou sinais adquiridos, eles tendem ser mais eficientes e podem ser treinados online e em tempo real.

Para aplicações em jogos, são desenvolvidos métodos que incorporam explicitamente um conhecimento prévio e a incerteza no processo de tomada de decisões, levando a métodos com a inferência Bayesiana, onde são os mais indicados para incorporar fontes distintas de medições produzidas com grande interferência e o conhecimento prévio incerto para o processo de diagnóstico.

Através desta pesquisa pode-se constatar, o frequente desenvolvimento da machine learning com foco em métodos lineares supervisionados e não supervisionados e a inferência Bayesiana, causando um grande impacto na assertividade das predições e jogadas. Nela será descrito algumas metodologias junto à exemplos da aplicação e as especificidades das jogadas de pôquer, com a finalidade de aprimorar o modelo.

Para a separação da fonte de dados às cegas, é necessário atentar-se à duas regras importantes para o aprendizado de máquina, a extração de dados estruturais é mais informativa do que os dados brutos em si e a inferência da estrutura de classes organizadas, o problema de classificação. No entanto, não é fácil separá-los para tratarmos em problemas distintos, considera-se os dois para ambos como métodos lineares.

Na mistura linear possui-se muitos casos onde o interesse está em separar ou fatorar um conjunto de dados observados em duas ou mais matrizes, utiliza-se de métodos padrões para decomposição de valores singulares (SVD) e a análise de componentes principais (PCA). Estes métodos conseguem atender critérios específicos de otimização, por exemplo PCA para termos de erros de reconstrução mínima, sobre a restrição de vetores de base ortogonais. Porém em muitos dos casos, os critérios não são muitos consistentes, quando dá-se pelo processo de formação de imagem e sinais e as matrizes resultantes possuem pouca relevância.

Supondo que um conjunto de dados observados é o resultado da combinação linear de fontes latentes, através desta mistura linear, é capaz de adquirir ou formar sinais e imagens subjacentes, consistente com uma mistura física, variando da eletroencefalografia. Conforme evidenciado por Paul Sajda (2006) dado X como uma matriz de observação M linhas por N colunas, a equação de mistura linear é dada por X = AS, onde A é o coeficiente de mistura e S a matriz de dados. Dependendo da modalidade as colunas X e S são as coordenadas do sistema na representação dos dados. O maior desafio é recuperar simultaneamente todos os dados em um único ponto de observação, este problema é denominado separação às cegas, pois a camada de dados não está diretamente visível e a matriz mista de dados não é conhecida. O método de busca às cegas é fundamentalmente aplicado a muitos problemas na recuperação de sinais.

A análise e classificação dos dados de jogos como o pôquer torna a tarefa um grande desafio, pois tudo que se deve fazer está cercado de incertezas além de possuírem muita interferência durante as jogadas, além do conhecimento prévio ser inconsistente, através da teoria de redes Bayesianas, abordando os princípios de inferência das propriedades dos dados incertos, tornou-se uma teoria popular na IA como um método de construção sistemas especialistas, pois a partir da sua representação evidencia a incerteza nos dados e no processo de tomada de decisão, métodos Bayesianos são capazes de resolver uma gama de problemas de inferência relevantes no campo dos jogos.

Juntamente com a incerteza de explicações, a teoria Bayesiana é capaz de diferenciar-se dos padrões e métodos de classificação, considerando-se a diferença entre modelos discriminatórios e generativos, ‘por exemplo modelo probabilísticos de reconhecimento ou discriminatórios, para isso é construído um modelo genérico, onde o modelo consegue estimar a partir de uma classe de distribuição, seguindo rigorosamente as regras de análise de dados. Esse modelo de classificação, por exemplo pode ser treinado utilizando o algoritmo de Bayes, obtendo a seguinte fórmula, conforme é explicado por Paul Sajda (2006) $P(C|D) = P(D|C)P(C)/P(D)$.

Além disso, novos exemplos relativos aos dados utilizados para o treinamento de dados podem ser identificados e serem computados a probabilidade sobre cada modelo. Essa capacidade de identificar novos modelos é útil para validar e estabelecer medidas de confiabilidade no sistema e nas suas saídas de dados, além da detecção dos novos dados servirem para treinar o sistema e torná-lo cada vez mais eficiente e refinado, acumulando experiência, reduz a vulnerabilidade à falhas.

Outro algoritmo aplicável na interpretação dos dos exames de urina, junto com a metodologia de aprendizado de máquina é a árvore de decisão, onde consiste em um conjunto de regras, onde dados previamente classificados serão processados e treinados, passando pela árvore. Nela também devemos informar as variáveis necessárias para a análise do exame de urina, juntamente com os parâmetros que são possíveis de serem encontrados naquela variável, definindo assim um mínimo e um máximo, como por exemplo quando encontramos leucócitos na urina, pode ser encontrado na quantidade baixa, média ou alta, com ela faz-se necessário informarmos um range de valores para cada variável, através dos testes

que irão ser realizados, iremos poder verificar se através dela será a mais eficiente para a análise e diagnosticar uma doença provável à partir de todas as variáveis.

6 ALGORITMOS

Algoritmos, uma das primeiras disciplinas vista no curso de ciência da computação, serve de fundamento para o resto da vida de um cientista da computação, sempre presente no desenvolvimento de softwares, trata a proposta de solução para os problemas apresentados.

O primeiro passo para o desenvolvimento de um algoritmos é realizar a análise do problema proposto, definir e colocar de forma que seja compreensível e facilite o desenvolvimento em uma linguagem descritiva, esta linguagem que serve para formalizar o problema e descrever é chamada de algoritmo, em seguida, o algoritmo é traduzido para uma linguagem de programação.

A linguagem de programação é o que intermedia as máquinas e os humanos, é a forma que os programadores têm de interagir com as máquinas, compreensível aos humanos e interpretável pelas máquinas, diferente da linguagem que possui uma sintaxe restrita à linguagem que é tratada, o algoritmo é aplicável a qualquer linguagem, basta escrevê-lo para a linguagem que deseja roda-lo.

Através da compilação, o código que foi escrito em linguagem de programação é convertido para a linguagem de máquina, a compilação é realizada através do computador e um programa intermediário chamado de compilador.

Um algoritmo é a abordagem do problema através do programador, ele representa como agiu através do problema abordado e demonstra para as outras pessoas através do seu algoritmo desenvolvido.

Existem premissas para considerar um algoritmo, ele deve ser legível, um leigo precisa compreender do que se trata, alto nível, ser capaz de ser traduzido para qualquer linguagem de programação, não pode ser restrito a um sistema operacional ou programa, preciso, não pode haver ambiguidade na descrição do algoritmo, conciso, não pode exceder uma página, caso isso ocorra, deve-se decompor o problema em subproblemas gerando assim outros algoritmos, estruturado, ele precisa ser facilmente identificável.

6.2 REDE BAYESIANA

Thomas Bayes foi um sacerdote inglês que morreu em 1761, antes de publicar sua obra, onde continha a fórmula que é a base da teoria Bayesiana.

É uma técnica de classificação baseada no teorema de Bayes, utilizada para representar o conhecimento probabilístico. Um classificador Bayes assume que uma determinada característica presente em uma classe não está relacionada à outra característica presente na classe, por exemplo uma fruta pode ser considerada como uma maçã se possuir sua cor vermelha, ser redonda e ter aproximadamente 3 centímetros de diâmetro. Independente que essas características estejam atreladas entre si ou até mesmo dependam da existência de outras características, um classificador Bayes consideraria todas essas propriedades para contribuir de forma independente para a probabilidade que o fruto é uma maçã.

Segundo Nádia Padua de Mattos (2005), pode ser um grafo acíclico utilizado para representar dependências probabilísticas, através de variáveis aleatórias ou contínuas onde cada nó representa uma variável aleatória associada a um objeto, possuindo diversos estados, podendo assumir qualquer um dos estados, sendo que cada estado representa uma probabilidade, a representação daria por objeto x, estados x1,x2,....xn, xi é a capacidade de assumir qualquer estado, com a probabilidade fica P(xi).

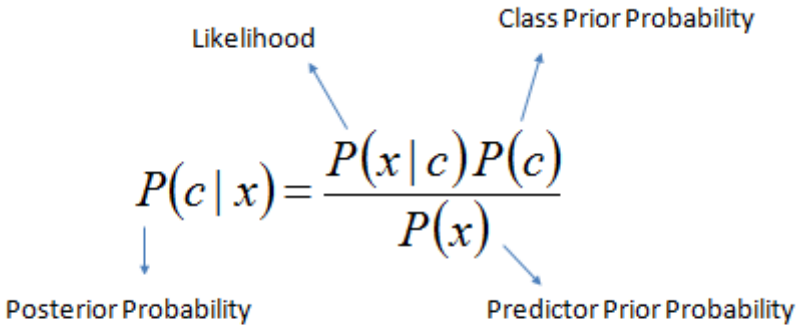
Através de V um conjunto finito de variáveis aleatórias definidas sobre o mesmo espaço amostral, P a distribuição de probabilidades sobre V, E um conjunto de relações binárias não reflexivas em V, e G = (V,E) um grafo acíclico direcionado. Cada vértice $v \in V$, seja $c(v) \subseteq V$, o conjunto dos pais de v, e $d(v) \subseteq V$, o conjunto formado pelos nós descendentes de v, para $v \in V$, seja $a(v) \subseteq V$ o conjunto $V – (d(v) \cup \{v\})$, ou seja, o conjunto das variáveis aleatórias de V excluindo v e seus descendentes. Supões que cada subconjunto $W \subseteq a(v)$, W e v, são independentes dado c(v), ou seja, se P(c(v)) > 0, isto é (Nádia Padua, 2005):

$$P(v \mid W, C(v)) = P(v \mid c(v)).$$

Então, C = (V, E, P) é denominada uma rede Bayesiana, também chamada de rede causal. Através dessa definição vimos que uma rede Bayesiana desbrava a independência condicional entre as variáveis do domínio que representa.

A seguir pode-se visualizar a representação da equação (1) do teorema de Bayes.

Figura 2 – Fórmula da probabilidade.



$$P(c \mid \mathrm{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c) \quad (1)$$

Fonte: Ray (2015)

Onde:

P (c | x) é a probabilidade posterior da classe (alvo) dados previstos (atributo).

P (c) é a probabilidade prévia de classe.

P (x | c) é a probabilidade que é a probabilidade de previsão da classe dada.

P (x) é a probabilidade prévia de previsão.

Para compreensão seguimos o exemplo encontrado no Analytics Vidhya, abaixo possui-se um conjunto de dados de treinamento de clima, correspondentes à variável alvo ‘Play’, o primeiro passo é converter os dados a uma tabela de frequência, em seguida cria-se a tabela de probabilidades, procurando a probabilidade ‘Overcast’ = 0,29 e a probabilidade dessa situação ocorrer é 0,64, conforme demonstrado na Figura 3.

Figura 3 - Tabelas de probabilidade

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

Likelihood table				
Weather	No	Yes		
Overcast		4	=4/14	0.29
Rainy	3	2	=5/14	0.36
Sunny	2	3	=5/14	0.36
All	5	9		
	=5/14	=9/14		
	0.36	0.64		

Fonte: Ray (2015)

Em seguida iremos utilizar a equação de Naive Bayes para calcular a probabilidade posterior, para cada classe, a classe com maior probabilidade posterior é o resultado da previsão.

Dado o problema: Os jogadores irão pagar se o tempo estiver ensolarado, está afirmação está correta?

Para resolvermos este problema utilizaremos o método discutido acima, para P(Sim | Sunny (Ensolarado)) = P(Sunny | Sim) * P(Sim) / P(Sunny).

Temos para P (Yes | Sunny) = 0.33 * 0.64 / 0.36 = 0.60. Com a maior probabilidade.

Naive Bayes utiliza um método similar de previsão da probabilidade de diferentes classes, baseado em diversos atributos, este algoritmo é mais utilizado na classificação de texto e problemas de múltiplas classes.

Nas redes Bayesianas temos a quantificação de incertezas através das tabelas de probabilidade condicionais conforme vimos logo acima. Condições que especificam a incerteza relacionada a um modelo representado nas redes são:

- Especificar as probabilidades a priori das hipóteses para as raízes da rede, cuja soma deve ser 1 para cada nó da raiz;

- Especificar as TPC's para os demais nós; cada linha da matriz é a distribuição condicional das hipóteses de um vértice dada uma realização conjunta dos nós pais; com isso, a somatória dos elementos de cada linha deve ser igual a 1.

A rede Bayesiana possui dois modelos, o probabilístico distribuindo um conjunto de probabilidades, este representa o conhecimento incerto e o segundo é

visto como mais relevante no processo de inferência em redes Bayesianas, onde ela é vista como uma linguagem.

Na inferência de redes Bayesianas, existem 2 tipos de computação que são executadas, a atualização de crença e a revisão de crença.

6.3 KNN

Primeiramente é necessário compreender o algoritmo NN, também conhecido como regra dos vizinhos mais próximos (k-nearest neighbor), foi proposto por Cover e Hart em 1966, técnica de fácil implementação e de conceitos fáceis de serem compreendidos. Para realizar a classificação de padrões divide-se em duas fases, a primeira é o treinamento dos dados obtidos através da técnica de *holdout* e a segunda são os testes o qual utiliza-se a base de testes também obtidas pela técnica *holdout*.

A fase de treinamento e testes dos dados é simples, dada uma base de dados para treinamento TRn={X1, ... , Xn}, onde n é o número de padrões armazenados em TR, é armazenado na memória todos os padrões de TR. Dado um padrão qualquer que pertença à base de treinamento ou de testes, possui-se duas características, um vetor de valores e uma classe. Durante os testes é analisado o desempenho do classificador para a classificação dos novos padrões. Com a base de testes TEn={X1, ... , Xn}, onde a classe de cada padrão é dada por Xn ∈ TE é conhecida e n é o número de padrões contidos em TE. Para a taxa de erros é necessário calcular através da equação seguinte.

Figura 4 – Fórmula de cálculo para taxa de erro.

$$\text{Tx. de Erro} = 100 \times \frac{\text{N.º de Padrões Classificados Incorretamente}}{\text{N.º de Padrões do Conjunto de Teste}}$$

Fonte: Eugenio 2005

Para que se calcule a taxa de erro do conjunto de testes, é necessário que o algoritmo encontre um padrão para cada erro, onde X ∈ TE n ∈ um padrão X ∈ TR '∈ , de forma que X ' dentre todos os padrões possua a menor distância Euclidiana. Durante os testes deve-se atentar ao algoritmo se ele cometeu algum erro de classificação do padrão, e ao final calcular a taxa de erro através da equação citada acima.

A partir do algoritmo NN surgiram diversas variações e a mais comum e utilizada é o KNN, surgindo com o objetivo de melhorar o desempenho de classificação, propondo modificações durante a fase de testes e classificação. Onde a proposta do algoritmo é dada por utilizar o K-vizinhos mais próximos, invés de utilizar os vizinhos mais próximos como era proposto no algoritmo original.

Para a classe de um novo padrão X, o algoritmo KNN calcula K-vizinhos mais próximos a X e classifica-o como a classe que possui a maior frequência dentre os K-vizinhos. Durante a classificação do algoritmo, às vezes ocorre um problema, onde o padrão X e seus K-vizinhos mais próximos são da mesma classe, e o algoritmo não consegue com qual classe dos K-vizinhos ele deve comparar o padrão X.

Segundo Webb (2002), KNN é um método simples de estimar a densidade local de padrões, de treinamento da vizinhança de um padrão desconhecido durante a classificação. Se Xk é o K-ésimo vizinho de X, então V será uma esfera centrada em X e com raio igual à distância Euclidiana entre X e Xk, ou seja, ||X- Xk||. O volume V da esfera, de raio r em n -dimensões, é dado pela equação Figura 5 a seguir.

Figura 5 – Fórmula para cálculo da densidade de padrões.

$$V = \frac{2r^n \pi^{n/2}}{n\Gamma(n/2)}$$

Fonte: Webb 2002

Onde Γ(x) é a função gama que é dada pela equação abaixo.

Figura 6 – Função gama

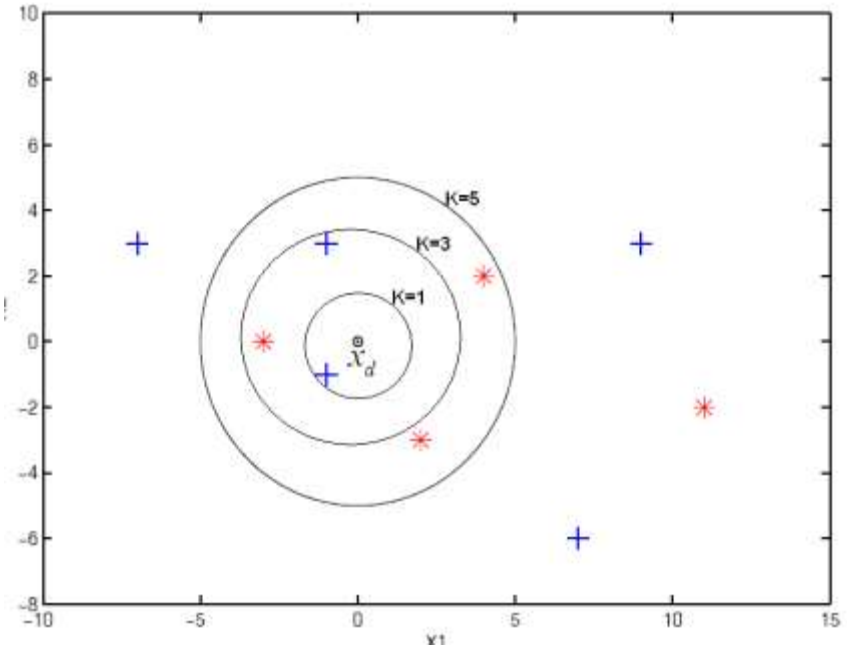
$$\Gamma(n+1) = \int_0^{\infty} x^n e^{-x} dx$$

Fonte: Webb 2002

O algoritmo possui o parâmetro K, onde através dele é realizado uma classificação mais refinada, porém o valor lógico de K varia de um problema para o outro, assim para cada base de dados é necessário realizar diversos testes para definir atribuindo valores diferentes a K a fim de descobrir qual é o melhor valor de K para o problema.

Um exemplo de classificação através do algoritmo KNN pode ser visto na figura abaixo. Onde Xd é o padrão desconhecido, pertencentes a uma das classes exibidas no exemplo, dependendo do número K-vizinhos, Xd será classificado da seguinte maneira, se K=1, Xd será classificado como “+”; se K=3, Xd será classificado como “+”; se K=5, Xd será classificado como “*”. (EUGÊNIO, 2005).

Figura 7 – Classificação de padrões.



Fonte: Eugênio (2006)

Classificação feita pelo KNN para um padrão desconhecido Xd.

6.4 ENSEMBLE LEARNING

É complicado traçar uma linha do tempo, e definir onde iniciou-se os métodos de aprendizado baseados em *ensemble*, existe uma ideia que pode ter sido visto múltiplos modelos de aprendizados, utilizados por muito tempo, porém a informação mais clara no assunto dá-se no início de 1990, onde é citado o *ensemble learning* em pesquisas. A primeira pesquisa publicada a respeito do assunto foi coordenada por Hansen e Salamon, terminando em 1980, onde eles encontraram previsões feitas através da combinação de um conjunto de classificadores mais aprimorados nas previsões, feitos por um único classificador bom. A segunda teoria encontrada na pesquisa de Schapire em 1989, onde ele prova que *weak learners* (aprendizes fracos), podem ser aprimorados através de *strong learners* e a prova resulta no aumento, esta é uma das maiores influências nos métodos *ensemble*.

Ensemble learning (Aprendizagem em conjuntos) ou também conhecido como *committee-based learning (*aprendizagem baseada em comitês), Multiple classifier systems (Classificador múltiplo de sistemas), Classifier combination (Classificador de combinações), visto que os termos comumente são encontrados em inglês.

O *ensemble learning* é uma vertente do aprendizado de máquina, como se fosse um dos formatos de aprendizado, como por exemplo aprendizagem supervisionada. Onde consiste em múltiplos aprendizes à serem treinados para resolverem o mesmo problema, ao contrário do que a aprendizado de máquina tradicionalmente aborda, tentando aprender com apenas uma única hipótese treinando dos dados, já os métodos do *ensemble* visam tentar construir um conjunto de hipóteses e combiná-las, e capazes de serem utilizadas.

No *ensemble learning* existe um número de aprendizes, popularmente chamados de *base learners (aprendizes bases)*, a *habilidade de* generalização do *ensemble learning* é muito mais poderosa que o *base learners*. O *ensemble learning* vem tornando-se cada vez mais utilizado, pois através da sua capacidade de potencializar os chamados *weak learners* de forma suave em aprendizes melhores, do que tentar procurar aleatoriamente os chamados *strong learners* que podem realizar previsões muito precisas. Sendo assim, os chamados *base learners* também são referidos aos *weak learners*, vale ressaltar, no entanto, embora a maioria das análises teóricas funcionem em aprendizes fracos, os *base learners* utilizados na

prática não são necessariamente fracos, através do uso de *base learners* não tão fracos, geralmente é obtido um resultado melhor no desempenho.

Usualmente os *base learners* são gerados em treinamento de dados a partir de dados obtidos através de algoritmos de aprendizagem de base, tais como árvore de decisões, redes neurais ou outro tipo de algoritmo relacionado à aprendizado de máquina. A maioria dos métodos encontrados no *ensemble* utilizam um único algoritmo de aprendizado de base, para a produção de *base learners* homogêneos, porém existem métodos capazes de gerarem aprendizes heterogêneos, esses por sua vez baseiam-se em algoritmos de aprendizagem múltipla. Em último caso, como não existe um único algoritmo de aprendizagem base, portanto as pessoas preferem chamá-los de aprendizes individuais (*individual learners)* ou aprendizes de componentes (componente learners) invés de simplesmente chamar de *base learners*. Os nomes *individual learners* e *componente learners* podem ser utilizados também em aprendizes homogêneos.

A construção de um *ensemble* é dada por dois passos. Primeiro, são produzidos vários *base learners*, que podem ser gerados em dois estilos, em paralelo ou em sequencial, onde a geração de um aprendiz tem influência na geração dos aprendizes subsequentes, então os *base learners* são combinados para serem usados, onde as regras de combinação é dita através dos esquemas de combinações mais populares, votando pela média ponderada e classificação, para a regressão.

Geralmente para ter um bom *ensemble*, os *base learners* devem ser os mais precisos possíveis e também os mais diversos. Assim eles possuem maior efetividade no processo de estimar a precisão dos aprendizes. Mas não segue-se uma definição rigorosa onde qual é o mais intuitivo percebido como diversificado. Embora diversas unidades de medidas tenham sido projetadas, essas medidas de diversidade na construção de um *ensemble* é considerada suspeita por Kuncheva e Whitaker. Na prática a diversidade de *base learners* pode ser introduzida à partir de diferentes canais, como por exemplo subamostrar os exemplos de treinamento, manipular os atributos, manipular as saídas, injetar aleatoriedade em algoritmos de aprendizado ou mesmo utilizar diversos mecanismos ao mesmo tempo. A utilização de diferentes processos básico de geração de aprendizes ou esquemas de combinação, leva a diferentes métodos do *ensemble.*

Existem diversos métodos efetivos no *ensemble*, a seguir serão explicados brevemente três métodos, *Boosting, Bagging* e *Stacking*. A classificação binária é considerada por simplicidade, isto é, X e Y, deixando eles indicarem o espaço da instância e o conjunto de classe, assumindo Y = {-1, +1}. Um conjunto de dados de treino D = {(x1, y1), (x2, y2), ..., (xm, ym)} é dado, onde xi ∈ X e yi ∈ Y (i=1, ..., m).

Boosting é um dos algoritmos mais famosos, ele possui diversas variações. O *AdaBoost* será considerado como exemplo. Primeiramente ele atribuiu o peso igual a todos os dados de treinamento utilizados como exemplo, a distribuição dos pesos na t-ésima rodada de aprendizado como Dt, do conjunto de dados de treinamento e Dt o algoritmo gera o *base learner ht : X → Y* chamado de algoritmo de aprendizagem base, então é utilizado o exemplo treinado para realizar os testes ht, e os pesos dos exemplos, classificados incorretamente irão aumentar, portanto, ocorre uma atualização na distribuição de pesos Dt+1. A partir do treinamento no conjunto de dados e Dt+1 o algoritmo *AdaBoost* gera um novo *base learner* chamando o algoritmo de aprendizagem base novamente, esse processo é repetido por T vezes, onde cada uma dessas repetições é chamada de rodada, e o último *learner* é derivado de através da votação por maioria ponderada dos *base learners* de T, onde o peso dos *learners* é determinado durante o processo de treinamento. Na prática o processo do algoritmo de *base learning* pode ser um algoritmo de aprendizado o qual consegue utilizar treinamento ponderado com exemplos diretamente, outro modo, os pesos podem ser explorados através de exemplos de um simples treinamento, de acordo com a distribuição Dt, a seguir é possível verificar o pseudocódigo do algoritmo AdaBoost na Figura 8.

Input: Data set $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_m, y_m)\}$;
Base learning algorithm $\mathcal{L}$;
Number of learning rounds T.

Process:
$D_1(i) = 1/m.$ % Initialize the weight distribution
for $t = 1, \cdots, T$:
$h_t = \mathcal{L}(\mathcal{D}, D_t);$ % Train a base learner h_t from $\mathcal{D}$ using distribution D_t
$\epsilon_t = \Pr_{i \sim D_i}[h_t(\boldsymbol{x}_i \neq y_i)];$ % Measure the error of h_t
$\alpha_t = \frac{1}{2}\ln \frac{1-\epsilon_t}{\epsilon_t}$; % Determine the weight of h_t

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \exp(-\alpha_t) & \text{if } h_t(\boldsymbol{x}_i) = y_i \\ \exp(\alpha_t) & \text{if } h_t(\boldsymbol{x}_i) \neq y_i \end{cases}$$

$$= \frac{D_t(i)\exp(-\alpha_t y_i h_t(\boldsymbol{x}_i))}{Z_t}$$

% Update the distribution, where Z_t is a normalization
% factor which enables D_{t+1} to be a distribution

end.

Output: $H(\boldsymbol{x}) = \text{sign}\left(f\left(\boldsymbol{x}\right)\right) = \text{sign}\ \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})$

Figura 8 – Pseudocódigo AdaBoost
Fonte: Zhi-Hua Zhou (2009)

* Versão traduzida pelo autor:

Entrada: conjunto de dados, algoritmo de aprendizagem base, número de rodadas de aprendizagem.

Processo: Inicializa a distribuição de pesos, treinando o aprendiz base, utilizando a distribuição Dt, medindo o erro de ht, determinando o peso de ht, atualizando a distribuição onde Zt está em normalização, fator que habilita Dt+1 para a distribuição.

O algoritmo Bagging treina uma série de base learners onde cada um possui uma amostra diferente de bootstrap, chamando um algoritmo de aprendizagem base. Uma amostra bootsrap é obtida a partir de uma subamostragem do treinamento do conjunto de dados realizando trocas, onde o tamanho do conjunto de dados da amostra é o mesmo que o conjunto de dados de treinamento. Portanto a amostra bootstrap, é o mesmo que os dados treinados, podendo alguns aparecer e outros não, onde a probabilidade em que os exemplos apareçam é menos que 0,632, após obter os base learners, Bagging irá combinar com a maioria dos votos e a classe mais votada será prevista, o pseudocódigo do algoritmo Bagging é mostrado na Figura 9, é importante ressaltar que um dos algoritmos variantes do Baggin a Random Forest foi considerado um dos algoritmos mais poderosos atualizados.

Input: Data set $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_m, y_m)\}$;
Base learning algorithm $\mathcal{L}$;
Number of learning rounds T.

Process:
for $t = 1, \cdots, T$:
$\mathcal{D}_t = Bootstrap(\mathcal{D})$; % Generate a bootstrap sample from $\mathcal{D}$
$h_t = \mathcal{L}(\mathcal{D}_t)$ % Train a base learner h_t from the bootstrap sample
end.

Output: $H(\boldsymbol{x}) = \text{argmax}_{y \in \mathcal{Y}} \sum_{t=1}^{T} 1(y = h_t(\boldsymbol{x}))$ % the value of $1(a)$ is 1 if a is $true$ and 0 otherwise

Figura 9 – Pseudocódigo Bagging

Fonte: Zhi-Hua Zhou (2009)

Na implementação típica do algoritmo Stacking, o número de individual de aprendizes de primeiro nível é gerado a partir de dados do treinamento empregado em diferentes algoritmos de aprendizagem, esses aprendizes individuais combinados a um aprendiz de segundo nível, o qual é chamado de meta-learner, na Figura 10 pode-se observar o pseudocódigo, evidenciando que o algoritmo Stacking realiza uma relação com as informações e métodos misturados.

Input: Data set $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_m, y_m)\}$;
First-level learning algorithms $\mathcal{L}_1, \cdots, \mathcal{L}_T$;
Second-level learning algorithm $\mathcal{L}$.

Process:
for $t = 1, \cdots, T$:
$h_t = \mathcal{L}_t(\mathcal{D})$ % Train a first-level individual learner h_t by applying the first-level
end; % learning algorithm $\mathcal{L}_t$ to the original data set $\mathcal{D}$
$\mathcal{D}' = \emptyset$; % Generate a new data set
for $i = 1, \cdots, m$:
for $t = 1, \cdots, T$:
$z_{it} = h_t(\boldsymbol{x}_i)$ % Use h_t to classify the training example $\boldsymbol{x}_i$
end;
$\mathcal{D}' = \mathcal{D}' \cup \{((z_{i1}, z_{i2}, \cdots, z_{iT}), y_i)\}$
end;
$h' = \mathcal{L}(\mathcal{D}')$. % Train the second-level learner h' by applying the second-level
% learning algorithm $\mathcal{L}$ to the new data set $\mathcal{D}'$

Output: $H(\boldsymbol{x}) = h'(h_1(\boldsymbol{x}), \cdots, h_T(\boldsymbol{x}))$

Figura 10 – Algoritmo Stacking

Fonte: Zhi-Hua Zhou (2009)

6.5 GRADIENT BOOSTING REGRESSOR

O Gradient Boosting Regressor é um módulo adicional e aprimorado encontrado no scikit-learn. Ele baseia-se no algoritmo Boosting e também no Regression, onde ele é capaz de realizar a construção de um modelo aditivo de forma progressiva, permitindo a otimização das funções arbitrárias de perdas diferenciáveis, onde cada etapa uma árvore de regressão se encaixa no Gradient negativo da função de perda.

Nela encontra-se diversos parâmetros para que possamos rodá-la, definimos o quanto teremos de perda, o valor de aprendizagem, o valor para performance de aprendizagem e quanto será aplicado o Boosting, o valor máximo de profundidade que o algoritmo irá atingir, critérios de avaliação e qualidade, valor mínimo em que os exemplos serão quebrados em nós internos, valor mínimo que os exemplos serão atribuídos a folhas, dentre outros parâmetros que não são obrigatórios para o funcionamento da função.

6.6 K-MEANS

O algoritmo K-Means é um algoritmo não supervisionado, um dos mais simples para resolução de problemas que envolvem clustering. O procedimento dá-se através de uma maneira fácil e intuitiva na classificação dos dados através de um determinado grupo fixados previamente, onde k é grupos.

Primeiramente deve-se definir um k central para cada grupo, e esse local de k deve ser muito bem escolhido, pois para cada local o resultado produzido será diferente, para a melhor escolha basta colocar um mais longe do outro. Em seguida define-se os dados do espaço ao k central mais próximo, quando acaba-se os dados para associação, finalizando esses passos temos um agrupamento inicial. Recalcula-se o k central, para novos centrais, sendo os baricentros dos grupos resultantes do passo anterior, após o cálculo é realizado uma nova associação de dados ao k central mais próximo. Gera-se um laço de repetição, com esse laço notamos que os k centrais alteram suas localizações, passo a passo até que cesse a mudança. A meta desse algoritmo é minimizar uma função objetiva, no caso uma função de erro quadrática.

Abaixo observa-se a função (2) objetiva.

Figura 11 – Cálculo função objetiva.

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} ||x_i^{(j)} - c_j||^2 \quad (2)$$

Fonte: https://www.ime.usp.br/~arcjr/machine/

Onde $||x_i^{(j)} - c_j||$ é uma medida de distância escolhida entre o ponto x_i e o centro do grupo c_j, é um indicador da distância entre os n dados do espaço de entrada e os seus respectivos centros de grupos.

7 ENGENHARIA DE SOFTWARE

Engenharia de software é uma das disciplinas dos cursos de computação, segundo o dicionário Aurélio define:

Informática: Ciência que visa ao tratamento da informação através do uso de equipamentos e procedimentos da área de processamento de dados.

Ciência: Conjunto organizado de conhecimentos relativos a um determinado objeto, especialmente os obtidos mediante a observação, a experiência dos fatos, um método próprio.

Processamento de dados: Tratamento dos dados por meio de máquinas, com o fim de obter resultados da informação representada pelos dados.

Engenharia: Arte de aplicar conhecimentos científicos e empíricos e certas habilitações específicas à criação de estruturas, dispositivos e processos que se utilizam para converter recursos naturais em formas adequadas ao atendimento das necessidades humanas.

Conforme citado acima, a informática é definida como ciência, onde seu assunto é o processamento de informações através de máquinas, já a ciência tem seu foco no acúmulo de conhecimento, através de métodos como experimentação e observação.

> Atendimento das necessidades humanas – O foco da engenharia é a necessidade humana. Nisto, ela tem escopo bem diverso da ciência. O conhecimento é certamente uma necessidade humana, mas uma entre várias outras de uma hierarquia: alimentação, moradia, segurança, afeição, auto-estima... Todo produto de engenharia se justifica através da satisfação de uma destas necessidades; portanto, da geração de algo que tenha valor para alguém. A Engenharia de

> Software procura gerar valor através dos recursos de processamento de informação. (Wilson, 2000).

O desenvolvimento de um software é dado a partir de um projeto, onde primeiramente faz-se reuniões junto ao cliente solicitante, compreende-se as necessidades e problemas que eles enfrentam, e a partir desses problemas apresentados pelo cliente, é feito uma análise do que será automatizado pelo software e como o software viabiliza e facilita as atividades diárias de quem utilizará e da corporação que ele irá atender.

Após a análise realizada junto ao cliente, dá-se o início do desenvolvimento do projeto, primeiramente é elaborada a documentação do projeto, determinando prazos, entregas, necessidades e demais informações que foram coletadas durante as entrevistas e análises, em seguida é realizado a elaboração de fluxogramas do sistema, também é desenvolvido diagramas, como o diagrama de UML, diagramas de classes, diagrama do banco de dados, telas do sistema para análise do cliente.

Nos dias atuais, tem-se falado muito de desenvolvimento ágil, projetos ágeis, e isso é aplicado, durante a elaboração dos projetos, desde a concepção junto ao cliente até o momento do desenvolvimento e por fim a análise e aceite do solicitante. Tem sido de grande ajuda as empresas desenvolvedoras de software a implementação de métodos ágeis e também ao próprio desenvolvedor, pois assim ele tem um maior controle da execução das atividades, uma maior quantidade de entregas referente ao projeto e a diminuição de riscos e custos do projeto com o retrabalho, pois as entregas são em menores prazos, com as parciais do sistema, onde o cliente consegue avaliar módulo a módulo, sem a necessidade de uma correção muito grande quando houver, por isso métodos ágeis vem sendo adotado cada vez mais no desenvolvimento de projetos.

Para o projeto, será utilizado o *kanban*, uma ferramenta que gerencia as atividades que serão desenvolvidas, organizando as atividades da seguinte forma em colunas, não iniciadas, fazer hoje, em andamento, em teste, realizada, impedido, cancelada, cada atividade passa por uma dessas colunas do *kanban*, com isso temos a noção de onde cada atividade está, como está a evolução delas e do projeto ao todo.

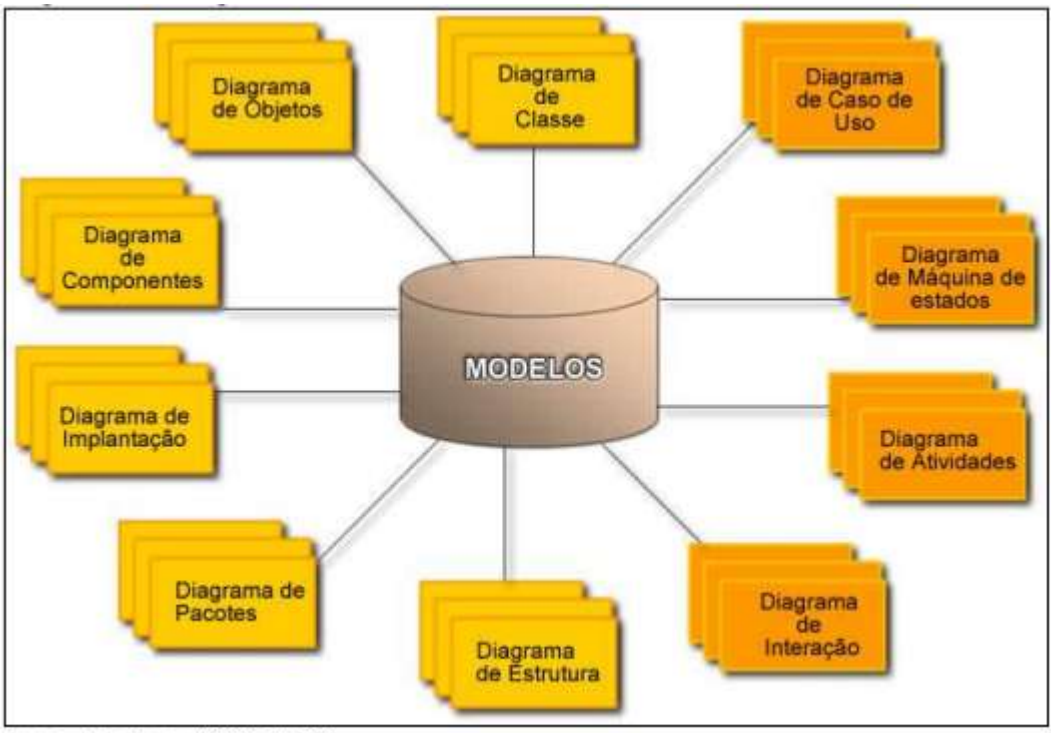
Para modelagem utiliza-se a linguagem de modelagem unificada, mais conhecida como UML, de acordo com Guedes (2004), é uma linguagem visual que visa modelar sistemas computacionais através de diagramas.

> A UML (Unified Modeling Language) é uma linguagem-padrão para a elaboração da estrutura de projetos de software. Ela poderá ser empregada para a visualização, a especificação, a construção e a documentação de artefatos que façam uso de sistemas complexos de software. (BROOCH; JACOBSON; RUMBAUGH, 2006, p. 13).

Existem 10 diagramas da UML, onde não existem dependências entre eles e cada um expressa uma função distinta e exclusiva.

Os diagramas são divididos em estruturais e comportamentais, conforme a Figura 9, abaixo.

Figura 12 – Diagramas de UML.



Fonte: Martinez (2006-2016).

Fowler (2005) caracteriza esse diagrama como de estrutura, comportamento e interações.

Diagrama de Caso de Uso é um dos mais utilizados para o desenvolvimento de projetos.

Para Guedes (2004) a sua aplicação é feita a fim de realizar consultas nas etapas de levantamento de requisitos, mas podendo ser utilizado nas etapas subsequentes do projeto.

Segundo Bezerra, E. (2015), as etapas inicias de um projeto dão-se para a criação das visões de caso de uso, esse recurso serve para dar uma visão externa do desenvolvimento do projeto e direcionar o ponto de vista de cada, e gerar interações aos agentes que causam interações.

Os dois autores, ressaltam que o diagrama de caso de uso é muito importante para o início do projeto, pois direciona e alinha com demais diagramas que possam vir a ser desenvolvido.

Diagrama de Classes, é considerado um dos mais importes por a maioria dos autores, sendo um dos mais utilizados para a elaboração de qualquer projeto que dá-se por desenvolvimento de software, servindo de apoio para demais diagramas no decorrer do projeto.

Guedes (2004, p.27) define “Como o próprio nome diz, define a estrutura das classes utilizadas pelo sistema, determinando os atributos e métodos possuídos por cada classe, além de estabelecer como as classes se relacionam e trocam informações entre si.”.

Conforme o autor fica claro que a UML é representada independente de qualquer linguagem de programação com qual o sistema será desenvolvido, sem que haja a necessidade de adaptação.

Para a etapa de testes de software, caracterizada como a revisão das funcionalidades, validação dos dados resultantes.

> Por melhores que sejam as técnicas de modelagem e especificação de software, por mais disciplinada e experiente que seja a equipe de desenvolvimento, sempre haverá um fator que faz que o teste de software seja necessário: o erro humano. É um mito pensar que bons desenvolvedores, bem concentrados e com boas ferramentas serão capazes de desenvolver software sem erros. (Beizer, 1990, citado por Wazlawick, 2013, p. 289).

Com os testes o desenvolvedor pode validar seu software, verificando se os processos estão coerentes, funcionais e com a usabilidade de acordo com quem irá utilizar no dia a dia.

Para os testes, será utilizado em grande parte o TDD (*Test Driven Development*) que significa desenvolvimento baseado em testes, testes estes que são escritos antes mesmo da produção dos próprios códigos, visando assim anteciparmos os testes e conferirmos nosso código enquanto desenvolvemos, pois através dele conseguimos gerar o código baseado nos testes pré-definidos, os códigos não ficam engessados com os pré-testes, caso haja algum teste que tenha sido esquecido, ele é escrito no decorrer do desenvolvimento juntamente com o código, garantido assim que o código passará pelos testes escritos.

O TDD é baseado em métodos ágeis, visando assim primeiramente a agilidade de entregas sem perder a qualidade do desenvolvimento, além de possuir um ganho na velocidade, tem-se também um ganho em relação à feedbacks da nova funcionalidade implementada, facilitando depois a refatoração do código.

Temos alguns ganhos utilizando o TDD no desenvolvimento do projeto, como por exemplo:

a) Feedback mais rápido sobre a nova funcionalidade.
b) Código mais limpo.
c) Segurança no refactoring.
d) Segurança na correção de bugs.
e) Maior produtividade no desenvolvimento, pois encontra-se menos bugs.
f) Código mais flexível.

8 FERRAMENTAS DE DESENVOLVIMENTO

Neste tópico serão apresentadas algumas tecnologias que servirão para desenvolver o trabalho.

8.2 PYTHON

Foi definido Python como linguagem de programação para a elaboração do projeto, pois é uma linguagem de fácil compreensão e utilização, não demanda de servidores potentes, existemuita documentação acerca da linguagem e sua utilização na elaboração de algoritmos voltados para o aprendizado de máquinas.

O Python está em várias frentes de trabalho, podendo ser utilizado para o desenvolvimento de ferramentas web, análise de dados, construção de aplicativos, construção de sistemas desktops, inteligência artificial, machine learning dentre outras aplicações.

O que motivou a escolha da linguagem para o desenvolvimento da pesquisa, foi a sua abrangência e por ser uma linguagem multiplataforma.

Ela conta com uma área focada na parte científica, onde existe um grande apoio em fóruns e na internet para quem está utilizando o Python com esta finalidade. Para o meio científico, existem dois grandes módulos que se associam ao Python, a Anaconda e o *Scikit-Learn*.

A Anaconda é um gerenciador de pacotes de fontes abertos voltados para a ciência, possuindo mais de 720 pacotes e podendo ser instalado em qualquer sistema operacional, tais como, Windows, macOS e Linux.

O Scikit-Learn é um módulo de código aberto, utilizado para a mineração e análise de dados a partir dos métodos de *Machine Learning*.

8.3 DOCKER

Visando facilitar ainda mais o desenvolvimento da aplicação e ter maior segurança, estabilidade durante o desenvolvimento e após o *start* da aplicação, foi escolhido o Docker, para manter a aplicação.

Docker é um *container* de plataformas, ele elimina a necessidade de utilizarmos as nossas máquinas para desenvolver uma aplicação e depois testá-la, ele funciona basicamente como se fosse uma VM (*Virtual Machine*), mas de maneira mais simplificada e consumindo bem menos da máquina que o hospeda, ele vem para solucionar problemas de desenvolvimento colaborativo, onde existe uma equipe de desenvolvimento por trás de um projeto.

As empresas estão utilizando Docker para o desenvolvimento de softwares ágeis, pois é uma ferramenta que facilita e agiliza o processo de criação de ambientes para desenvolvimento entre outras atividades, proporcionando entregas mais ágeis dos projetos, pois além de ser uma ferramenta de fácil utilização, eles conseguem criar novos ambientes sem muita burocracia ao contrário de quando se trabalha com uma VM, além de evitarem problemas que normalmente ocorrem com a VM, com o Docker é possível trabalhar lado a lado em *container* isolados, sem a influência um dos outros, diferente do que ocorre com uma VM compartilhada normalmente.

A diferença do *container* para as VM, é que ele não utiliza um sistema operacional para sustentá-lo, e depois é instalado um sistema operacional acima dessa camada, igual acontece com as máquinas virtuais.

Através do Docker, tem-se a facilidade de trocar de máquina, onde está se desenvolvendo uma aplicação, sem a complexidade da VM.

8.4 PROGRESSIVE WEB APPS

Progressive Web Apps é uma tecnologia desenvolvida pela Google, a tecnologia visa que não seja mais desenvolvido aplicativos nativos, onde esse aplicativo seja simples e não irá utilizar nenhum recurso nativo do celular, eles recomendam que invés de desenvolver um App nativo, desenvolva um PWA.

Primeiramente o PWA, vem para facilitar o desenvolvimento de aplicativos, pois ele é baseado em HTML e CSS, utilizado para o desenvolvimento de sites, a diferença que com o PWA, você possui manifestos e recursos que possuem acesso à alguns recursos do celular, como por exemplo a notificação *push.*

Os PWA são mais rápidos, segundo a documentação do Google Developers, são 53% mais rápidos que apps nativos, visto que para o carregamento de um PWA, é necessário apenas os recursos necessários e os demais recursos são armazenados na memória cache do celular.

Muitos questionam: já que o PWA é basicamente um site, ele terá de ser utilizado on-line? Não, pois através dos manifestos, existem formas de armazenarmos o que é necessário para o funcionamento do PWA offline, semelhante à um app nativo, o PWA só irá consumir internet no momento que houver uma requisição para o servidor, para trazer alguma informação, semelhante o que ocorre com os apps nativos.

O PWA tem as seguintes premissas, confiável, rápido e cativante. Onde cada uma delas tem sua descrição dada pelo Google:

a) Confiável - Carrega instantaneamente e nunca mostra problemas na rede, mesmo em condições adversas.
b) Rápido - Responde rapidamente as interações do usuário.
c) Cativante - Sinta-se como um aplicativo nativo do aparelho, com a experiência imersiva do usuário.

9 METODOLOGIA

O objetivo deste trabalho, foi elaborar um protótipo de jogador autônomo capaz de jogar pôquer Texas Hold'em no Limit. Tornando capaz provar que é possível ter jogadores autônomos em uma mesa de pôquer virtual, sendo capaz de realizar decisões de jogadas semelhantes ao de um ser humano.

Afim de apresentar a capacidade dos algoritmos de aprendizado de máquina e até onde eles podem chegar.

O jogador autônomo foi desenvolvido em linguagem Python, a partir das bibliotecas modelo disponibilizadas pelo MIT, juntamente com scikit-learn, numPy e sciPy. Para implementar o algoritmo foi escolhido o Gradient Boosting Regressor, pois durante o estudo dos modelos, foi possível identificar que para esse tipo de aprendizado, o algoritmo escolhido teve uma melhor performance.

Foram realizados testes e comparações com os algoritmos citados no trabalho, comparando a performance do KNN, K-Means, Naive Bayes em relação ao Ensemble Learning utilizando Gradient Boosting Regressor, quando determinado que seria utilizado este algoritmo, foi dado início o treinamento massivo do jogador, para depois concluirmos sua eficácia.

No princípio foi transferido a base de dados, tentando gerar uma versão dela em MongoDB, mas notou-se que não seria viável no futuro a utilização do mesmo.

Outro teste realizado, durante os experimentos e tentativas no desenvolvimento do trabalho, foi através da criação de containers utilizando o Docker, para verificar diferentes versões dos algoritmos e treinamentos.

Durante o desenvolvimento do jogador autônomo, e testes dos algoritmos, foi realizado o TDD a fim de validar todas as condições impostas, validando o funcionamento da aplicação.

Para que o trabalho fosse executado, através de uma base de dados disponibilizado pelo MIT, com algumas demos de treinamento, foi executado o algoritmo para realizar o treinamento dos dados a fim de verificar o resultado no final.

10 RESULTADOS

Foi realizado diversos treinamentos e comparações, e concluiu-se que o algoritmo Gradient Boosting Regressor, sai-se melhor que os demais testados, como KNN, K-Means, Naive Bayes, para este tipo de aplicação.

Visto que o algoritmo tem a capacidade de realizar regressões no seu aprendizado e optar pelo melhor aprendiz, visando obter sempre o melhor conjunto de aprendizes, para o treinamento do jogar, essa prática foi excelente, pois trata-se da aleatoriedade do baralho de cartas utilizada no jogo e também a incerteza de que qual carta cada adversário pode ter, além das cartas que serão postas à mesa. Tratando assim, de um aprendizado semelhante ao do ser humano diante de algo novo.

Na demonstração a seguir, podemos notar como foi a partida de pôquer, com os jogadores que estavam em treinamento, é possível observar que o jogador realiza as negociações dos valores do pote com os demais, suas cartas que estão em mãos e as cartas que foram postas na mesa.

Mão 5

Player 2(1141) negociou 8d and Qs

Player 4(59) negociou 6c and As

Player 2 apostou no small blind

Player 4 apostou big blind

Player 2 chamou a próxima carta 1 vez

Player 4 aumentou de 11 para 13

Player 2 aceitou 11

['3h', 'Kc', '5c']

Player 4 confere.

Player 2 aumentou para 270

Player 4 all-in

Retornou para o Player 2

['3h', 'Kc', '5c', 'Qd']

['3h', 'Kc', '5c', 'Qd', '8s']

Player 2 vence.

Player 2 foi treinado com mais de 10 mil mãos que o Player 1, conforme é exibido na Figura 10.

Na figura podemos notar, a diferença que um jogador sobressai ao outro, em relação aos treinamentos, quanto mais treino, mais mãos jogadas, melhor a performance dele, esta é uma das vantagens do algoritmo utilizado.

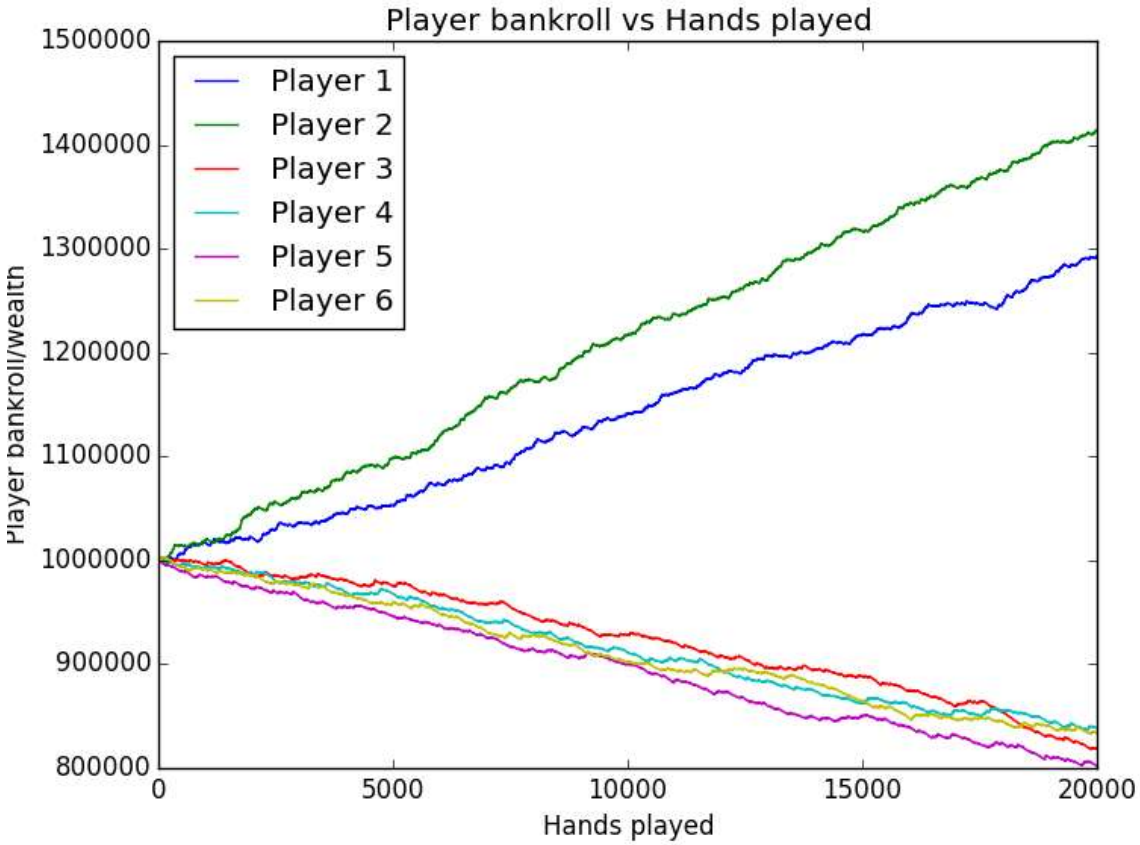


Figura 10: Obtida através das execuções do Scikit-learn.

11 CONSIDERAÇÕES FINAIS

De acordo com a pesquisa realizada, pode-se verificar que é possível implementar um jogador autônomo para a prática do pôquer também, além de outros jogos. Verificando o modelo disponível, ser semelhante ao modelo utilizado por uma equipe do MIT, conseguindo tornar-se campeã em um campeonato de pôquer, utilizando apenas um modelo de ensemble learning.

Foi possível notar a qualidade e autonomia, do algoritmo empregado durante o aprendizado, obtendo um jogador capaz de ser implementado em outras aplicações, como um bot, capaz de jogar em mesas de pôquer online ou até mesmo um jogador professor, que ensina pessoas que tenham interesse em aprender a jogar pôquer.

Porém, o pôquer é um jogo que consiste em muitas variáveis, tais como ambiente, comportamento do adversário, análise física dentre outras, o modelo pode ser constantemente aprimorado, levando sempre a uma melhora, para assim obter os melhores resultados.

12 REFERÊNCIAS

ANALYTICS VIDHYA. **Basics of ensemble learning**. Disponível em: <https://www.analyticsvidhya.com/blog/2015/08/introduction-ensemble-learning/>. Acesso em: 18 ago. 2017.

CROSS VALIDATED. **Baggin, boosting, and stacking**. Disponível em: <https://stats.stackexchange.com/questions/18891/bagging-boosting-and-stacking-in-machine-learning>. Acesso em: 18 ago. 2017.

CS NJU EDU CN. **Zhi-hua zhou homepage**. Disponível em: <https://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/springerebr09.pdf>. Acesso em: 17 ago. 2017.

DOCKER. Docker documentation. Disponível em: <https://www.docker.com/>. Acesso em: 18 mai. 2017.

FELIPE MOJAVE. **Você conhece sobre odds?**. Disponível em: <http://felipemojave.com/>. Acesso em: 01 jul. 2017.

GOOGLE DEVELOPERS. Web. Disponível em: <https://developers.google.com/web/progressive-web-apps/>. Acesso em: 06 abr. 2017.

MIT POKER BOTS. **Poker bots**. Disponível em: <http://mitpokerbots.com/>. Acesso em: 18 ago. 2017.

MIT TECHNOLOGY REVIEW. **An ai poker bot has whipped the pros**. Disponível em: <https://www.technologyreview.com/s/603544/an-ai-poker-bot-has-whipped-the-pros/>. Acesso em: 20 ago. 2017.

OREGON STATE UNIVERSITY. **Cs or**. Disponível em: <http://web.engr.oregonstate.edu/~tgd/publications/mcs-ensembles.pdf>. Acesso em: 18 ago. 2017.

RAY, S. Essentials of machine learning algorithms (with python and r codes). **ANALYTICS VIDHYA,** 2015. Disponível em: <https://www.analyticsvidhya.com/blog/2015/08/common-machine-learning-algorithms/>. Acesso em: 18 mai. 2017.

RUSSEL, S.; NORVIG, P.. **Artificial intelligence**: a modern approach. 3 ed. [S.L.]: Prentice Hall, 2009. 1152 p.

SCIKIT-LEARN. **Ensemble methods**. Disponível em: <http://scikit-learn.org/stable/modules/ensemble.html>. Acesso em: 20 ago. 2017.

SHARMA, Pragya; KUMAR, Deep. Comparative Analysis of KNN and C5.0 Algorithm for Smart City Classification. **International Journal of Engineering and Technical**

Research, [S.L], v. 7, n. 4, p. 54, abr./mai. 2017. Disponível em: <https://www.erpublication.org/published_paper/IJETR2151.pdf>. Acesso em: 18 mai. 2017.

THE MIT PRESS JOURNALS. **Boosting regressior estimators**. Disponível em: <http://www.mitpressjournals.org/doi/abs/10.1162/089976699300016746>. Acesso em: 18 ago. 2017.

TOPTAL. **Ensemble methods**. Disponível em: <https://www.toptal.com/machine-learning/ensemble-methods-machine-learning>. Acesso em: 18 ago. 2017.

WINSTON, P. H.. **Artificial intelligence**. 3 ed. [S.L.]: Pearson, 1992. 737 p.