
# Utilizando Aprendizado de Máquina para predição de resultados da NBA

**Caio Regueira Graça Melo**

**Relatório do Projeto Final de Graduação**

**Centro Técnico Científico – CTC**
**Departamento de Informática**
Curso de Graduação em Ciência da Computação

Caio Regueira Graça Melo

Utilizando Aprendizado de máquina para predição de resultados da NBA

**Relatório** de Projeto Final II, apresentado ao programa de **Ciência da Computação** da PUC-Rio como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Marco Serpa Molinaro

Rio de Janeiro, Novembro de 2021

**Resumo**

Este projeto tem a finalidade de utilizar diversas estatísticas envolvidas na NBA em uma variedade de modelos de aprendizado de máquina. A partir dos resultados, estamos propondo uma utilização dos modelos em casos no mundo real, utilizando de exemplo a casa de apostas.

**Palavras Chave**

NBA, Aprendizado de máquina, Redes Neurais

## Sumário

# 1. Introdução

Predição é uma das muitas aplicações para o aprendizado de máquina. Utilizando diferentes modelos e algoritmos, é possível prever diversos cenários e resultados. O basquete, um dos esportes mais populares do mundo, possibilita múltiplas aplicações para o aprendizado de máquina, entre eles a predição de resultados e estatísticas.

A escolha do basquete não está somente atrelada a popularidade da liga e do esporte. A NBA, liga americana de basquete que foi essencial para a popularização do esporte, também é conhecida por ser uma liga composta de diversas estatítiscas, tendo a disposição milhares de dados que, se estudados e utilizados da melhor forma, podem resultar em excelentes modelos de predição.

Diante das diversas pesquisas envolvendo o tópico, o desenvolvimento de uma pesquisa comparativa onde serão aplicados modelos de aprendizado de máquina de complexidades variadas será relevante. Essa pesquisa teria o intuito de fazer predições focando no vencedor de confrontos da NBA e tentando prever a quantidade de pontos feita por cada time em uma partida. Além disso, uma análise extensa sobre os atributos selecionados seria um excelente complemento para a pesquisa.

O projeto consiste no desenvolvimento de um sistema que consiga realizar a seguinte predição: dado um jogo de basquete da NBA, prever qual das duas equipes será a vencedora, baseado nas informações sobre as equipes. Ademais, o projeto será complementado com uma análise das features selecionadas e uma aplicação dessas previsões no mundo real, mais especificamente utilizando as cotações (odds) do mercado de apostas para verificar o possível lucro obtido pelas predições dos modelos.

## 1.1. Situação Atual

Desde o início da utilização de aprendizado de máquina para predição de estatísticas/resultados esportivos, mais especificamente para a NBA, muitos pesquisadores e organizações tentam criar os melhores modelos possíveis, procurando atingir altas percentagens de acerto em suas predições. Dentre as pesquisas/modelos encontrados na literatura, a precisão média para predição de vencedor varia entre 60% e 70%, com modelos que atingem até 80% de precisão, utilizando ANNs (Artificial Neural Networks).

Quando colocamos em perspectiva a predição de resultados da NBA, o objetivo será prever não somente o vencedor de certo jogo, mas também quantos pontos cada time marcou naquela partida. É importante ressaltar que os modelos de predição citados acima focam majoritariamente na previsão de um vencedor de dado confronto.

Como indicado na seção anterior, a predição de resultados esportivos é um tópico muito popular, com muitos trabalhos disponíveis que podem servir como base de inspiração para este projeto. Por exemplo, temos o projeto realizado pela empresa Oursky [3], já citado acima, que mostra uma alta precisão para a tarefa de predição de um vencedor, mas não detalha tanto os atributos utilizados, se atendo principalmente ao discurso da utilização de uma abordagem baseada em times, jogadores e uma mistura do dois. Em outro exemplo, Johnson [4] realiza uma interessante análise de eficácia dos atributos, mostrando um gráfico de correlação dos atributos escolhidos, além de uma aplicação no mundo real através do mercado de apostas, no entanto ele se apega a atributos mais simples, tais como percentagens de arremessos, assistências, rebotes e etc, deixando de explorar atributos mais complexos que poderiam incrementar a performance de seu modelo, tais como o ELO [5] e atributos baseados em performance de jogadores. Weiner [6] e Vaidya [7] realizaram outros interessantes projetos relacionados a seleção de atributos: Vaidya mostra uma análise em relação a quais atributos podem ser significativos na predição de vitórias, e Weiner faz um acompanhamento bem completo da busca pelos melhores atributos, disponibilizando diversos gráficos que fazem comparações de um time em diferentes temporadas, tentando fazer uma ligação entre esses atributos e a performance do time.

## 1.2. Atividades Realizadas

Em relação ao ambiente de desenvolvimento, a familiarização com o IDE Spyder Anaconda foi feita nas aulas de computação aplicada, assim como as bibliotecas de Pandas e Numpy, cujos conhecimentos foram desevolvidos não somente nessa aula, mas também nas aulas de probabilidade computacional e computação para economia. Em relação à aprendizado de máquina, a introdução ao tópico foi realizada na aula de inteligência artificial, mas houve um estudo mais profundo no começo de 2021 quando a escolha do tema do projeto foi feita.

Como foi dito em seções anteriores, a escolha da liga de basquete americana foi feita por causa da variedade de estatísticas que a própria liga dispõe de forma

gratuita através de sua API, assim possibilitando a realização de diversas análises. A ideia inicial do projeto era realizar o procedimento de web scrapping do próprio site da NBA para extrair as informações diretamente do HTML. Web scrapping é o trabalho de extração dos dados presentes em um website. Caso não fosse achada uma API ou biblioteca que fossem gratuitas e que oferecessem auxílio nesse caso.

### 1.2.1. Primeira Etapa

O principal estudo que foi feito para o desenvolvimento do projeto foi o curso Machine Learning A-Z [8], na Udemy. Esse curso foi muito útil para o aprendizado da visão geral de aprendizado de máquina, além da lógica por trás de seus conceitos e da aplicação de seus principais modelos através da biblioteca SciKit-Learn. No curso, são ensinados conceitos bem básicos das bibliotecas Pandas e Numpy, que são essenciais para essa tarefa de ciência de dados. No entanto, muitos dos conceitos utilizados, principalmente no módulo de engenharia de atributos, foram pesquisados e estudados de acordo com a necessidade. Um exemplo desses conceitos é a detecção de streaks em um DataFrame [9], que foi necessário para calcular a “forma” dos times, ou seja, se eles vinham de uma série de derrotas ou vitórias.

### 1.2.2. Segunda Etapa

Como foi mencionado, a API utilizada para a coleta dos dados primários foi a API oficial da NBA. Nessa API, são oferecidos dezenas de endpoints, desde dados mais básicos, como estatísticas de um time em uma temporada ou em uma janela de tempo escolhida, até dados mais complexos, como box scores em tempo real, e informações em tempo real sobre a movimentação dos jogadores em quadra. Para decidir qual seria o endpoint usado, foi feito um estudo da documentação da API e uma exploração dos endpoints disponíveis, testando diversas funções com uma variedade de parâmetros. Após essa análise da API, os endpoints escolhidos para serem utilizados foram dois. Um para buscar todas as estatísticas dos times da NBA dada uma janela de temporadas escolhida pelo usuário, e outra para buscar todas as estatísticas dos jogadores da NBA para a mesma janela de temporadas. Além disso, foi necessário um estudo de como realizar o trabalho de web scrapping, que foi utilizado para extrair as cotações de todos os jogos adquiridos através da API. Esse trabalho foi feito utilizando as bibliotecas Selenium, que ajudou a automatizar a interação com o site escolhido para buscar as cotações dos jogos [22], e a

BeautifulSoup, que foi muito útil para buscar as informações necessárias diretamente do HTML do site.

### 1.2.3. Terceira Etapa

Para colocar os conhecimentos adquiridos em prática com os dados gerados nas etapas anteriores, foram desenvolvidos seis modelos de aprendizado de máquina com diferentes complexidades. A escolha de vários modelos foi importante para podermos visualizar a performance de cada um comparando uns com os outros, assim tendo uma melhor visão de qual é melhor e qual é o pior para o nosso contexto. Os modelos utilizados são: Regressão Logística, Kernel Support Vector Machine (SVM), Naive Bayes, Random Forest, Rede Neural Artificial e LSTM.

### 1.2.4. Quarta Etapa

Nessa última etapa, o foco foi aplicar os resultados adquiridos na etapa anterior em um cenário no mundo real, sendo ele o exemplo da casa de apostas. Para essa seção do projeto, os conhecimentos das bibliotecas Numpy, Pandas e Matplotlib foram essenciais, visto que foi necessário um grande trabalho de manipulação de dados para que pudessem ser gerados resultados propícios para a visualização.

### 1.2.5. Cronograma

No primeiro semestre o foco foi o estudo e aprendizado dos métodos e ferramentas a serem utilizados, além da redação dos relatórios necessários na primeira parte do projeto. Nessa primeira parte, os protótipos iniciais também foram utilizados para prever uma rodada na NBA que ocorria na época, acertando o vencedor de 7 dos 8 jogos no dia.

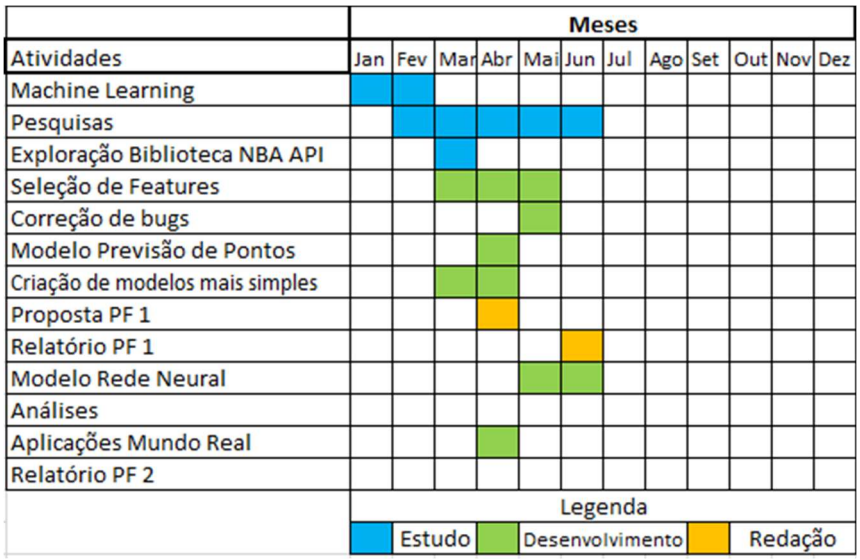



*Figura 1 - Cronograma Primeiro Semestre*

No segundo semestre a proposta inicial era desenvolver mais os modelos de previsão de pontos e, por fim, fazer as análises das aplicações das cotações. A principal diferença entre o previsto e realizado foi a falta de desenvolvimento dos modelos de regressão, que ficaram em segundo plano diante da aplicação das cotações, que se mostraram mais relevantes para o projeto. Além disso, também foi gasto bastante tempo no início do semestre para o melhor entendimento dos modelos de aprendizado profundo, mais especificamente do modelo de LSTM. Na imagem abaixo, temos o cronograma planejado para o segundo semestre e, logo em seguida, o cronograma executado nesse período.

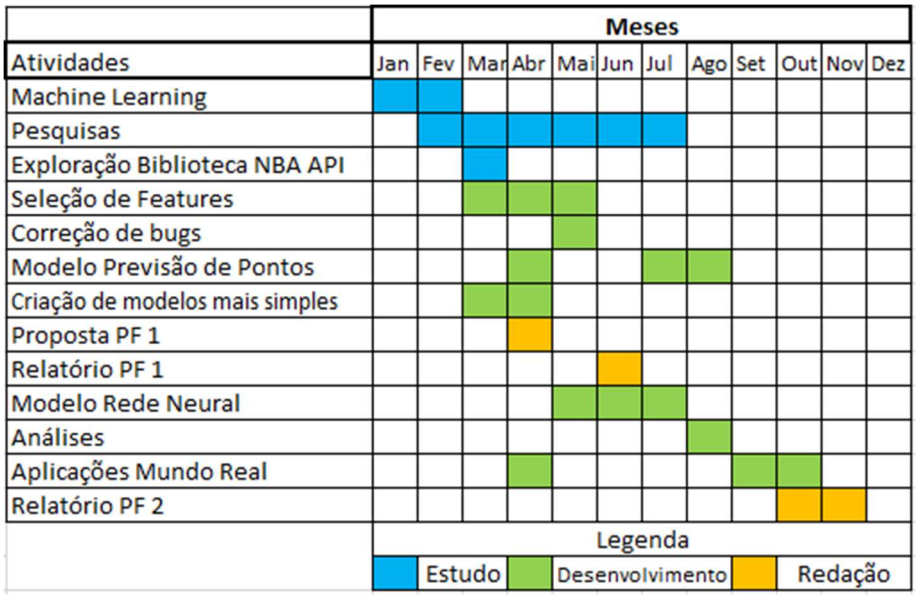



*Figura 2 - Cronograma Planejado Segundo Semestre*

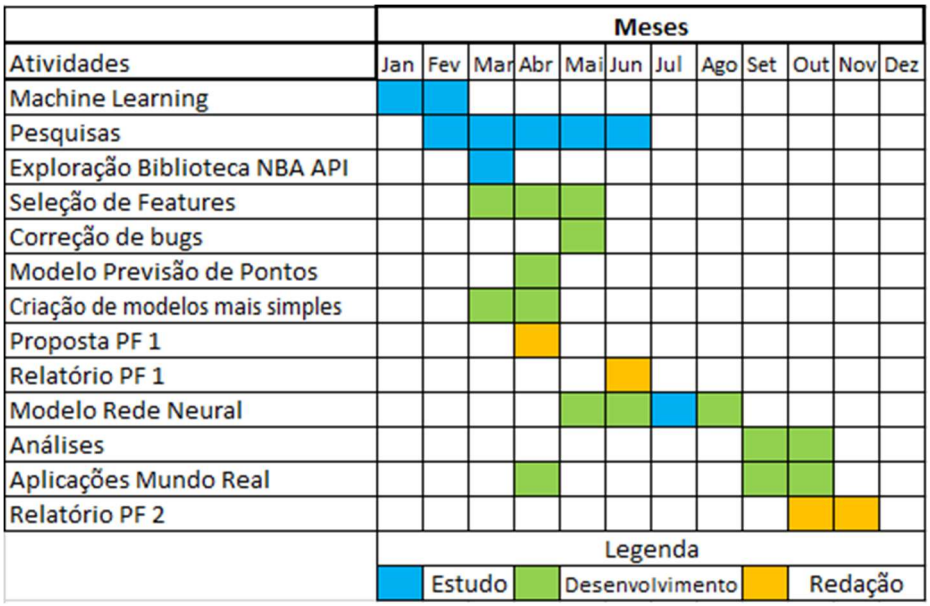



*Figura 3 - Cronograma Executado Segundo Semestre*

## 2. Modelos de Aprendizado de máquina

Para introduzir os modelos de aprendizado de máquina utilizados no projeto, é importante destacar como um modelo aprende a partir dos dados que lhe são fornecidos. Esse aprendizado ocorre pelas chamadas “Cost Functions”, ou função de utilidade. Elas são funções que determinam o quão bem (ou mal) está a performance de um modelo [12]. Geralmente esse resultado é gerado a partir da distância entre o resultado previsto e o resultado real. O objetivo desses modelos é tentar ao máximo minimizar os resultados gerados por essas funções, assim atingindo a menor distância possível entre os valores reais e previstos. Abaixo, está a ilustração do algoritmo de gradiente descendente, que busca otimizar a procura de um mínimo local ou global de uma função de utilidade.

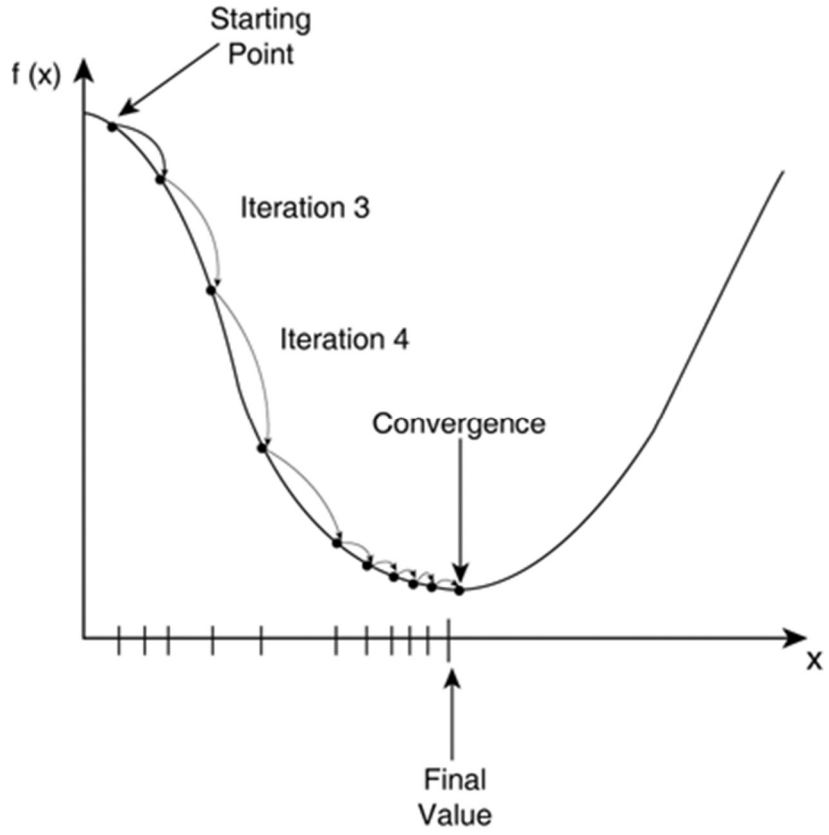



*Figura 4 - Gradiente Descendente [12]*

A seguir descreveremos brevemente os modelos de classificação que serão utilizados neste projeto.

## 2.1. Regressão Logística

Primeiro relembramos o modelo de regressão linear. Esse modelo consiste em atribuir pesos para as variáveis dependentes em uma equação linear. Nessa equação, destacada abaixo, temos o coeficiente “slope” para representar esses pesos, atribuído para cada variável e representado na equação abaixo por beta [25].

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

Na regressão logística, a ideia principal é a aplicação de uma função sigmóide na equação da regressão linear indicada acima [13]. A partir dessa aplicação, chegamos a um modelo onde podemos calcular a probabilidade de uma nova instância ter uma das duas classes (classificação binária), por exemplo, se o vencedor de uma partida foi o time A ou o time B, no nosso caso. Como são geradas

probabilidades, a forma do modelo categorizar um resultado é a partir da definição de um limiar, geralmente representado por 0.5. Dessa forma, se a probabilidade de um time A ganhar uma partida for menor que 50%, o modelo preverá que o time B será o vencedor do confronto.

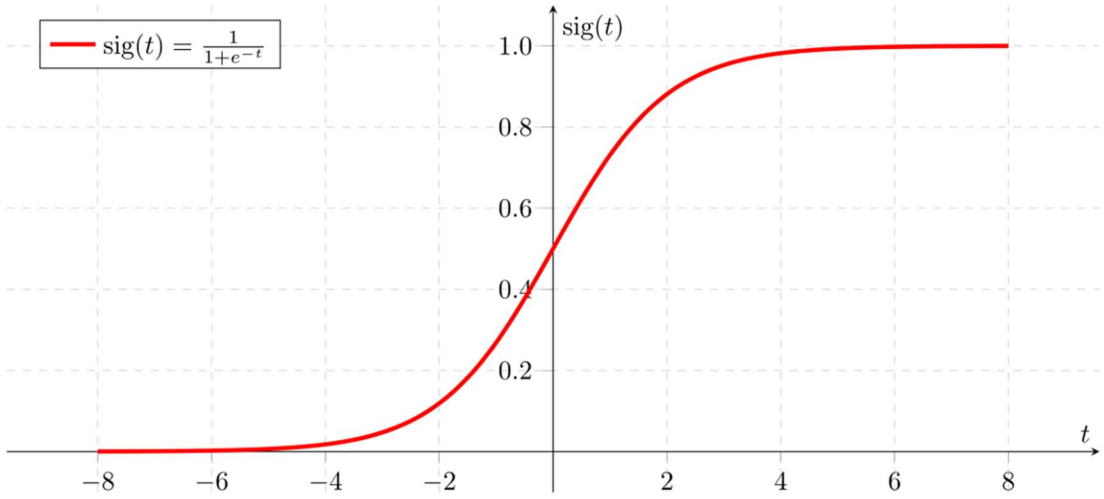



*Figura 5 - Função de Ativação Sigmóide [13]*

No caso da regressão logística, a utilização de uma simples função de custo de mean squared error não seria suficiente para esse modelo, visto que o algoritmo de gradiente descendente só consegue chegar ao mínimo global caso a função seja convexa.

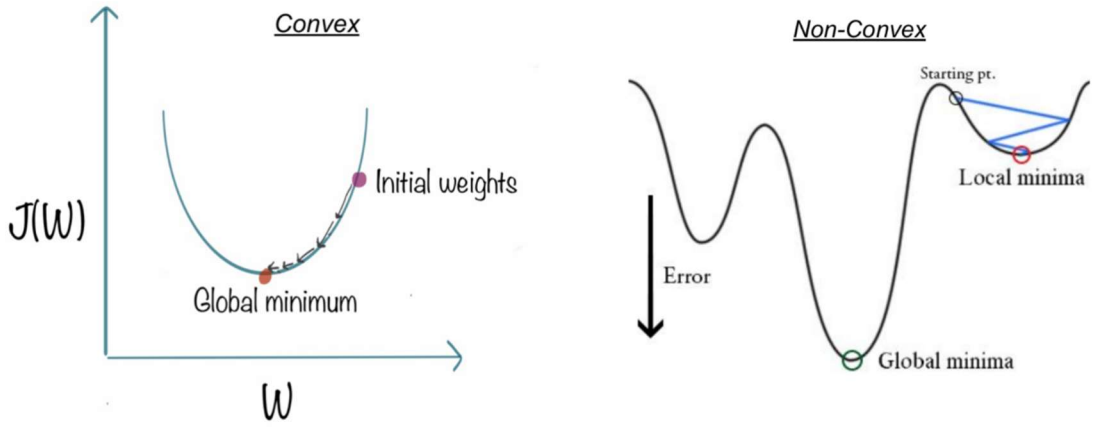



*Figura 6 - Gradiente Descendente em Funções Convexas e Não Convexas [14]*

Portanto, modelo utiliza a seguinte função de custo, construída a partir de funções logarítmicas, onde o $h_\theta(x)$ representa o valor previsto pelo modelo e o $Y(actual)$ representa o valor real, resultando em uma função final convexa:

$$\text{Cost}(h\Theta(x), Y(\text{actual})) = -\log(h\Theta(x)) \text{ if } y=1$$

$$-\log(1- h\Theta(x)) \text{ if } y=0$$

*Figura 7 - Função de Custo da Regressão Logística [13]*

## 2.2. Naive Bayes

O classificador de Naive Bayes se baseia na probabilidade de uma nova instância ter a sua classe sendo X ou Y a partir de instâncias antigas que possuem atributos compatíveis aos da nova instância.

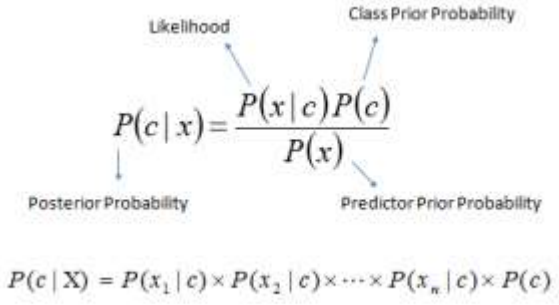



*Figura 8 - Fórmula de Naive Bayes*

Na imagem acima, podemos ver a fórmula que descreve o classificador de Naive Bayes. Nele, nós prevemos a probabilidade de itens cujos n atributos tem valor $x1,x2,...xn$ serem da classe "c". O principal passo dessa fórmula é, para todo atributo "i", obter a probabilidade de uma instância da classe c possuir tal atributo com um valor igual a "$xi$". Obtidas as probabilidades, temos que multiplicá-las, chegando ao valor $P(c \mid X)$, ou a probabilidade de uma instância da classe c possuir um conjunto de dados iguais a $X$. Em seguida, realizaremos uma multiplicação desse valor pela probabilidade de um conjunto de dados ser da classe "c" ($P(c)$) e, por fim, dividimos esse resultado pela probabilidade de um conjunto de dados ser semelhante à "X" ($P(x)$):

Dessa forma, temos a probabilidade de o conjunto de atributos em questão ser da classe "c". A classe prevista para o conjunto será a que possui a maior probabilidade.

## 2.3. Support Vector Machine

O modelo se baseia na separação das possíveis categorias através de divisores chamados “hiperplanos” [31]. Para tornar o conceito mais claro, podemos imaginar um modelo bem básico com somente dois atributos. Aplicando o SVM nesse modelo, teríamos uma linha que separa duas categorias, como é possível ver no exemplo abaixo.

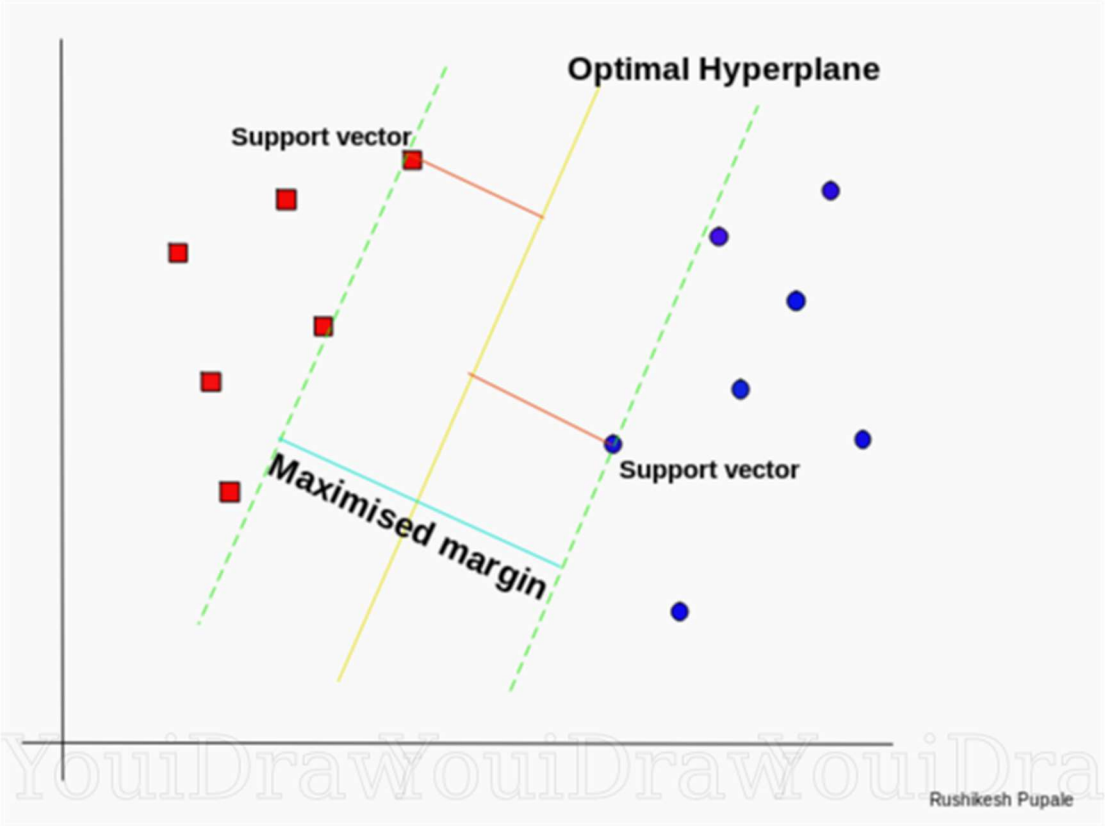



*Figura 9 - SVM [31]*

Quando utilizado em modelos mais complexos, esses os hiperplanos são construídos em um espaço multidimensional, tornando mais difícil a sua visualização. Para obter o hiperplano, nós buscamos o elemento de cada categoria que minimiza a distância entre os elementos de outras categorias. Assim que acharmos esses elementos, nós criamos um divisor entre eles que seja equidistante dos dois. As linhas que representam ambos os lados equidistantes ao hiperplano - as linhas pontilhadas do exemplo acima - são considerados os nossos vetores de suporte. Equivalentemente, isso corresponde a obter o classificador com a maior margem para os dados, veja a Figura 12. A partir dessa divisão, conseguimos classificar novas instâncias. O SVM pode ser estendido utilizando o conceito de kernel, chegando-se

no modelo de Kernel SVM. Esse modelo é caracterizado por utilizar um espaço dimensional maior, podendo aplicá-lo para dados não linearmente separáveis, veja a Figura 13 para ilustração.

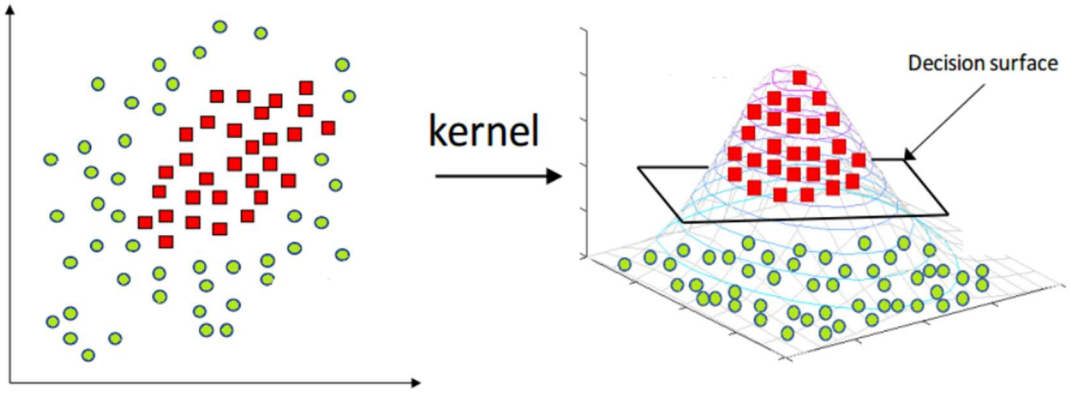



*Figura 10 - Kernel SVM [32]*

Quando os elementos não são perfeitamente separáveis, utiliza-se a hinge loss function [16].

## 2.4. Random Forest

O Classificador de Random Forest é um modelo que utiliza o conceito de ensemble learning, onde um algoritmo de Árvore de Decisão será executado múltiplas vezes, e suas predições combinadas em uma única predição. No algoritmo de Árvore de Decisão, conseguimos alcançar a definição da categoria de uma nova instância a partir de diversas perguntas binárias relacionadas aos seus atributos, resultando em uma árvore.

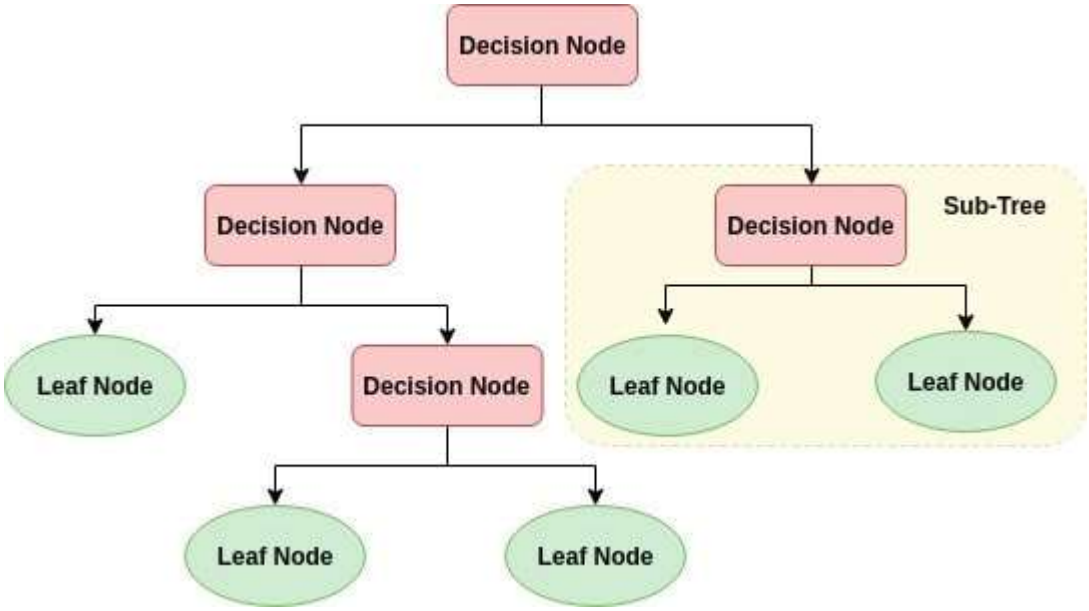



*Figura 11 - Árvore de Decisão [14]*

Uma funcionalidade interessante do classificador de random forest é a capacidade de gerar a importância de cada atributo para o modelo. Essa funcionalidade possibilitou a geração de um gráfico onde conseguimos analisar quais foram os atributos mais pertinentes no treinamento do modelo.

## 2.5. Redes Neurais

### 2.5.1. Rede Neural Artifical (Artificial Neural Network)

Nesse modelo, nós utilizaremos a noção de neurônios, que são basicamente nós que recebem sinais de entrada e geram sinais de saída. Nesse caso, os sinais de entrada virão inicialmente da chamada “input layer”, ou camada de entrada, onde cada nó representa um atributo independente. A partir dessa camada inicial, podemos ter diversas camadas conhecidas como “Middle layers”, ou camadas intermediárias, que é onde serão atribuídos pesos para que o modelo consiga aprender. Todos esses pesos serão computados por um neurônio e os sinais de saída serão baseados na aplicação de uma função de ativação nos pesos recebidos [19].

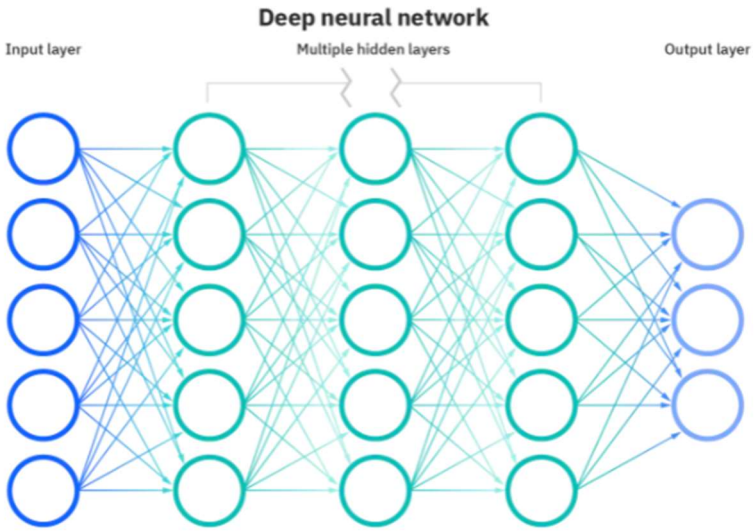



*Figura 12 - Rede Neural [19]*

No conceito simples, consideraremos que esses sinais de saída já sejam os valores previstos. Com esses valores, o modelo aplicará uma função de custo para comparar os valores previstos com os reais, e assim podendo diminuir o valor gerado por essa função a partir do balanceamento, ou tuning, dos pesos definidos anteriormente. Esse processo se chama “back-propagation” [20]. A partir desse processo, conseguiremos um modelo com balanceamento de pesos otimizado para a previsão de futuras instâncias.

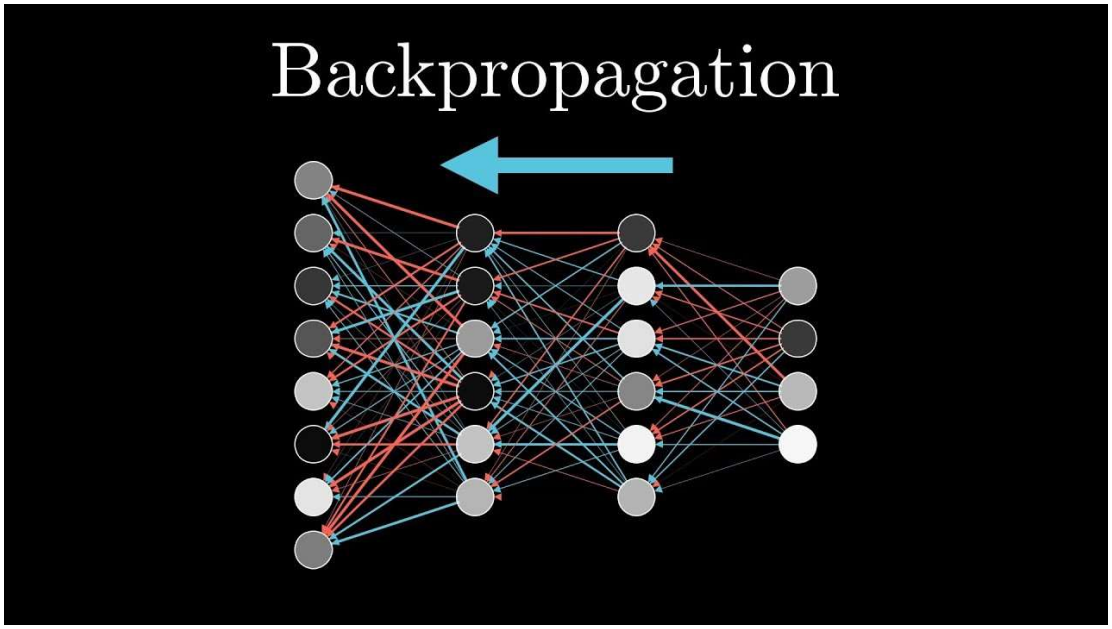



*Figura 13 - Backpropagation [20]*

### 2.5.2. Rede Neural LSTM

Redes neurais recorrentes são muito utilizadas para predição de dados sequenciais, como séries temporais. A definição desse conceito se baseia na capacidade da rede de armazenar informações para a utilização no longo prazo, caso elas se mostrem pertinentes na obtenção do resultado. Isso é possível por causa de loops que permitem a persistência de informações [21].

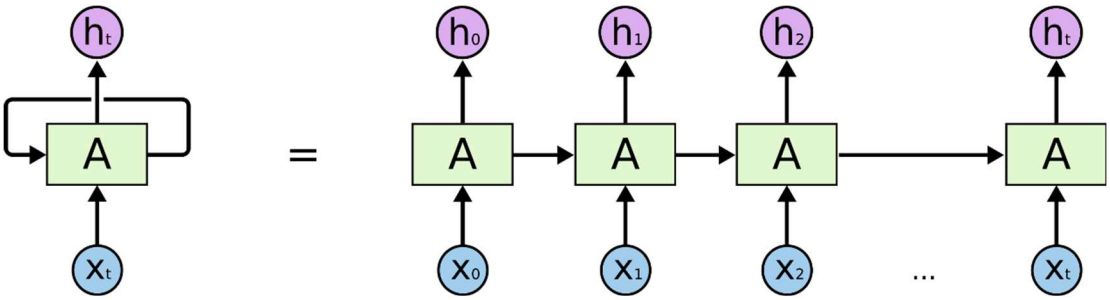



*Figura 14 - Rede Neural Recorrente [21]*

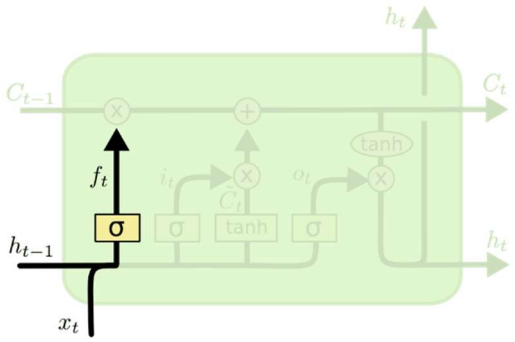
Na imagem acima, conseguimos ver que apesar de os nós possuírem uma entrada e saída definidas em X e H, respectivamente, eles também possuem uma seta apontando para o próximo nó. Essa seta permite carregar informações pertinentes que possam ajudar os nós posteriores em suas predições. Esse tipo de rede neural é muito utilizado em contextos de reconhecimento de fala, modelagem de idiomas, tradução, captura de imagens e entre outros.

O problema de uma rede neural recorrente está na capacidade para aprender e utilizar informações recebidas no passado. Em um contexto onde é necessário buscar um dado utilizado há pouco tempo para uma predição, uma RNN tem a capacidade aprender a usar tais informações. O problema se torna evidente quando é necessário buscar uma informação recebida em uma interação mais distante, mas que é necessária no contexto atual. Para isso foi criado o conceito da rede neural Long Short Term Memory (LSTM).

As LSTMs são um tipo especial de rede neural recorrente onde o seu diferencial é exatamente uma das deficiências de uma simples RNN, a capacidade de aprender dependências de longo prazo. Elas foram introduzidas por Hochreiter & Schmidhuber em 1997, sendo desenvolvida e evoluída por muitos outros autores ao longo dos anos.

Em relação a arquitetura de uma rede LSTM, é importante destacar o conceito de “gates”, que são estruturas que possuem o trabalho de remover ou adicionar informações ao estado da célula atual, ou seja, informações guardadas de módulos anteriores. Eles são compostos de uma camada sigmóide onde o output varia de 0 a 1, onde nenhuma informação passará para o estado da célula no caso de 0, e tudo será passado para o estado no caso de 1.

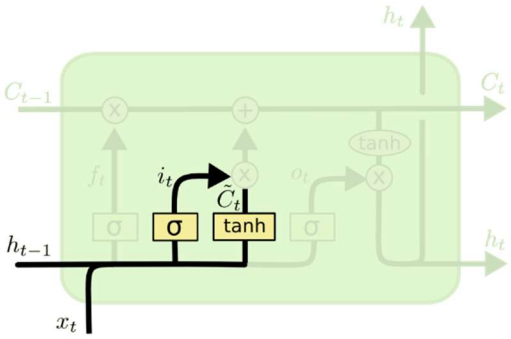
No primeiro passo da LSTM, é necessário definir o que vai ser mantido e o que vai ser esquecido do estado da célula. Isso é feito com a ajuda da aplicação de uma sigmóide no chamado “forget gate”.




$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

*Figura 15 - Forget Gate [21]*

O segundo passo consiste em definir o que será armazenado no estado da célula. Esse processo é divido em duas etapas. Primeiramente, devemos aplicar uma função sigmóide no chamado “input gate” para definirmos os valores que serão atualizados. Logo depois a camada que aplica a função tanh gera um vetor “Ct linha” com novos candidatos a serem adicionados ao estado da célula.



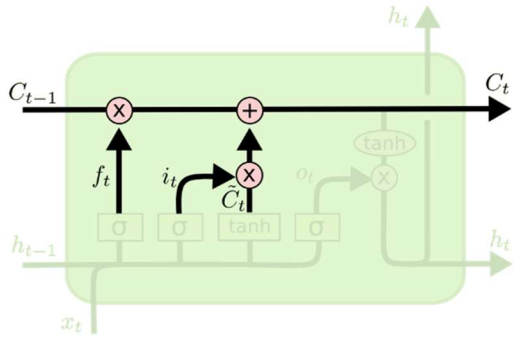

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

*Figura 16 - Input Gate [21]*

Após a execução desses dois passos, chegamos ao ponto em que devemos atualizar o estado anterior da célula “Ct-1” em um novo estado “Ct” a partir das

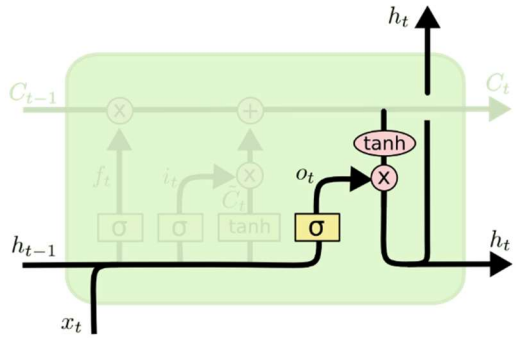
operações executadas em passos anteriores. Nesse passo, primeiramente é executada uma operação de multiplicação com “ft”, dessa forma esquecendo as informações definidas anteriormente no forget gate. Em seguida, as informações que foram escolhidas para serem mantidas e as potenciais novas informações a serem guardadas são multiplicadas e adicionadas ao estado atualizado “Ct”.




$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

*Figura 17 - Atualizando o Estado da Célula [21]*

Por fim, uma função sigmóide é aplicada para a obtenção do conjunto de dados escolhidos para saírem no output, e aplicamos uma tanh no estado da célula, multiplicando pelo resultado da sigmóide. Dessa forma, chegamos ao conjunto final do output.




$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

*Figura 18 - Output [21]*

Um ponto a ser ressaltado na implementação da LSTM está no formato dos dados de entrada. Diferentemente dos dados utilizados nos modelos anteriores, onde em uma mesma instância temos as informações das duas equipes antes de uma partida, o problema é visto como uma série temporal na LSTM, onde as estatísticas das N partidas prévias ao confronto no qual queremos prever são consideradas na entrada [35].Por causa disso, o formato final dos dados de entrada do modelo LSTM é o seguinte: (número de instâncias, número de N instâncias prévias para prever o próximo confronto, número de atributos em uma instância).

### 2.5.3. Dropout

Para evitar que ocorra um overfitting nos modelos de redes neurais, o método de dropout foi utilizado. Esse método consiste em remover neurônios aleatoriamente de uma interação, seja para frente ou para trás (back-propagation) [23]. A decisão se um neurônio é removido ou não é decidida de forma randômica através de uma probabilidade “p” de um neurônio ser mantido na camada. A utilização desse método faz a rede não ficar dependente de certos nós, evitando criar um padrão que pode torná-lo muito específico ao conjunto de treino, consequentemente causando o overfit.

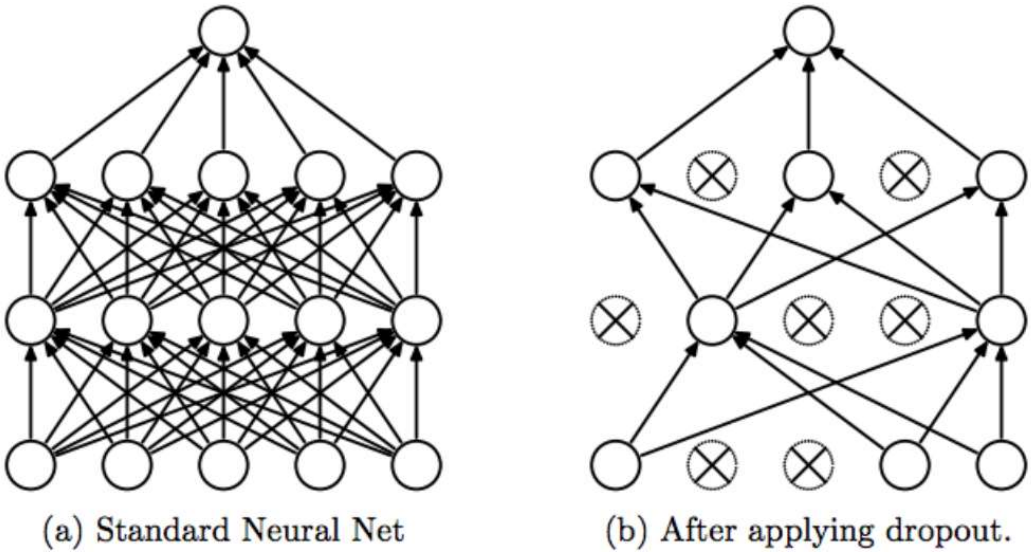



*Figura 19 - Dropout [23]*

# 3. Escolha de atributos para a previsão de partidas

Lembre-se que o nosso objetivo é o seguinte: dado uma partida entre dois times, prever se o vencedor será o time da casa ou o visitante. Como foi mencionado na introdução, a definição de um bom conjunto de atributos é essencial para a otimização da acurácia do sistema. Então uma parte importante desse projeto foi desenhar e coletar atributos relevantes para previsão de jogos de basquete.

Após a avaliação conceitual e avaliação experimental inicial, onde selecionamos manualmente as features que seriam utilizadas, chegamos ao seguinte conjunto de atributos:

- Pontos (PTS)
- Pontos concedidos (PTS_CON)

- Aproveitamento nos arremessos de quadra (FG_PCT)
- Aproveitamento nos arremessos de três pontos (FG3_PCT)
- Aproveitamento nos lances livres (FT_PCT)
- Rebotes (REB)
- Turnovers (TOV): Desperdícios de posse de bola
- Bloqueios (BLK)
- Aproveitamento nos jogos da temporada (SEASON_PCT)
- Aproveitamento dentro ou fora de casa (dependendo do contexto da próxima partida) (HA_PCT)
- ELO (ELO): Método estatístico para calcular a força de uma equipe [5]
- Streak da equipe (STREAK): Sequência do número de vitórias ou derrotas
- Aproveitamento nos últimos confrontos contra o adversário da partida em consideração (MATCHUP)
- Cotação para a vitória da equipe (ODDS): Extraídos da plataforma de cotações históricas Odds Portal [22]
- Média de PER dos jogadores previstos para jogar a partida (PER): método estatístico para calcular a contribuição de um jogador por minuto jogado [10]

Diante dos atributos mencionados acima, é importante ressaltar que eles são utilizados para ambos os times ao tentar prever o vencedor de uma partida. Além disso, todas as estatísticas que são próprias de uma partida, tais como pontos, rebotes, assistências e entre outros, são calculados a partir de uma média das últimas 10 partidas, assim chegando a uma noção de momento da equipe em relação àquele atributo específico. Dentre os atributos citados, os que merecem o maior destaque por serem um pouco mais complexas, são o ELO e o PER. Ambos são métodos estatísticos que visam a definição de força, seja de uma equipe no caso do ELO, ou de um jogador no caso do PER.

## 3.1. ELO

O ELO é um método criado pelo professor de física húngaro-americano chamado Arpad Elo. Arpad era um mestre do xadrez e seu método foi inicialmente criado para a quantificar a qualidade de um jogador de xadrez. Depois de sua criação, esse método passou a ser utilizado em diversos esportes como futebol, futebol americano, baseball, basquete e entre outros. A utilização do ELO no basquete teve

como seu principal agente a empresa FiveThirtyEight [11], uma empresa famosa por realizar projetos de aprendizado de máquina em diversas áreas, desde a previsão de campeonatos esportivos até a previsão de eleições presidenciais dos Estados Unidos. A empresa, inclusive, possui um gráfico interativo [29] onde é possível acompanhar o progresso do ELO de todos os times da NBA, indo de desde 1970 até hoje. Para calcular o ELO de uma equipe, primeiro definimos o ELO inicial de todos os times como 1500. Em seguida, é realizada uma iteração de rodada a rodada, onde o ELO de um time é atualizado ao final de cada partida que ele joga. Definimos essa atualização do ELO a partir da seguinte fórmula, onde vic representa 1 caso o time em questão tenha vencido, ou 0 caso tenha perdido:

$$k = 20 * (vicPtsMargin + 3)^{0.8} / (7.5 + 0.006 * winnerEloMarginDiff)$$

$$e = 1/(1 + 10^{(oppElo - teamE\ )/400})$$

$$ELO = k * (vic - e) + eloTeam$$

Ao final de cada temporada, o ELO de todos os times sofre uma pequena regularização a partir da seguinte fórmula:

$$ELO = ELO * 0.75 + 1505 * 0.25$$

## 3.2. PER

O PER, como nós já mencionamos acima, é um método que visa definir a produtividade de um jogador, ou seja, o quanto ele contribui por minuto jogado. Essa fórmula foi criada por John Hollinger, e sua sigla tem o significado de Player Efficiency Rating, ou classificação de eficiência do jogador. A fórmula tenta conseguir essa classificação se baseando em estatísticas positivas, tais como rebotes, roubadas de bola e assistências, e em estatísticas negativas, tais como turnovers e arremessos não convertidos. Para calcular o PER de um jogador, temos a seguinte fórmula:

$$\begin{aligned}[ FGM\ x\ 85.910\ +\ Steals\ x\ 53.897\ +\ 3PTM\ x\ 51.757\ +\ FTM\ x\ 46.845 \\ +\ Blocks\ x\ 39.190\ +\ Offensive_{Reb} x\ 39.190\ +\ Assists\ x\ 34.677 \\ +\ Defensive_{Reb} x\ 14.707\ -\ Foul\ x\ 17.174\ -\ FT_{Miss} x\ 20.091 \\ -\ FG_{Miss} x\ 39.190\ -\ TO\ x\ 53.897\ ] * (1/Minutes)\end{aligned}$$

Definida a fórmula de cálculo do PER, é feita a média dos 10 últimos jogos de cada jogador que está previsto para jogar a partida em consideração. Por fim, chegamos a média PER do time ao dividir o somatório da média PER individual de cada jogador pelo número de jogadores previstos para jogar a partida.

# 4. Experimentos Computacionais

Nessa seção realizaremos experimentos computacionais para verificar o poder de predição dos modelos. A tarefa é prever qual dos times de um confronto será o vencedor (mandante/visitante). Faremos também a avaliação dos atributos e avaliação do desempenho no cenário de apostas.

## 4.1. Ambiente Computacional

O sistema operacional utilizado no desenvolvimento do projeto é o Windows 10 Home, em um notebook com processador i5-8250U e 8GB de RAM. A linguagem de programação é o Python, linguagem mais utilizada para desenvolvimento de aplicações que usam aprendizado de máquina [1]. Isso ocorre muito por conta de sua simplicidade e extensa oferta de bibliotecas que auxiliam o desenvolvimento. Os ambientes utilizados são o Spyder Anaconda para a coleta, pré-processamento dos dados e execução dos modelos de aprendizado de máquina, assim como a realização da análise e visualização dos dados. A execução desses modelos será feita através das bibliotecas Scikit-Learn para modelos mais simples, e TensorFlow para o desenvolvimento de redes neurais. Além disso, serão utilizadas as bibliotecas Pandas e Numpy para a melhor manipulação dos dados, e as bibliotecas Matplotlib e Seaborn para uma melhor experiência na visualização dos dados.

A NBA, como disse anteriormente, é uma liga que oferece diversas estatísticas para uma análise, disponibilizando para desenvolvedores uma API que fornece diversos dados referentes ao seu campeonato. Para acessar essa API, é utilizada a biblioteca nba_api [2], que torna consideravelmente mais simples o acesso à API através da linguagem Python. Além dessa API, serão utilizadas as bibliotecas Selenium e BeautifulSoup para o scrapping de dados, a fim de obter as cotações históricas das partidas no site Odds Portal [22].

## 4.2. Modelos testados e seus parâmetros

Antes de entrar em detalhes nos resultados obtidos pelos modelos, listamos os modelos testados os parâmetros utilizados (quando nenhum parâmetro é mencionado, utilizamos o padrão da biblioteca Scikit-Learn).

- **Naive Bayes**

- **Rergessão Logística**
- **Kernel SVM**
- **Random Forest**. Para este algoritmo, utilizamos o algoritmo de força bruta RandomizedSearchCV [33], que teve um tempo de execução em torno de duas horas, para buscar os hiper parâmetros óptimos para a sua execução. Após a execução desse algoritmo, definimos o número de árvores de decisão presentes na floresta como 1000. Além disso, definimos o número mínimo de amostras para o nó ser considerado um nó folha como 2. O número de amostras necessárias para dividir um nó interno foi definido como 5, a altura máxima de uma árvore como 10 e bootstrap ativado. Por fim, o número de features necessárias para levar em consideração ao procurar uma melhor divisão foi definido como a raiz do número total de atributos.
- **ANN**. Para o modelo mais simples de rede neural, foram adicionadas três camadas ao modelo, sendo uma de entrada, uma intermediária e uma de saída. Para as duas primeiras camadas, o número de neurônios definido foi de 68, com a função de ativação “relu”, ou função de retificador [34]. Por fim, na última camada, foi utilizada a função sigmóide para realizar a classificação. Além disso, para tentar evitar o overfitting, foram aplicados os valores 0.001 de regularização e 0.9 de dropout. Para o treinamento da ANN, foi utilizado um batch size de 32, com 100 epochs.
- **LSTM**. Para essa rede foram utilizadas somente duas camadas, uma de entrada e uma de saída. Na camada de entrada, foram utilizados 12 neurônios, além da descrição do formato de entrada (input_shape) descrito na Seção 2.2.5.2. Ademais,para evitar o overfitting, foi definido o valor 0.9 de dropout para a camada de entrada. O batch size e número de epochs são os mesmos utilizados na ANN.

Além desses algoritmos também adicionamos aos experimentos dois baselines, ou seja, métodos simples que servem como base de comparação para os outros métodos. Utilizamos a **baseline de confrontos** e a **baseline de odds**. Na baseline de confrontos, que é a mais simples, o vencedor previsto é o que mais ganhou em confrontos anteriores contra o adversário em questão, enquanto na baseline de odds,

o vencedor previsto será àquele com as menores cotações, indicando que ele é favorito no confronto de acordo com as cotações históricas da Odds Portal.

## 4.3. Acurácia dos modelos e análise de atributos

Para a obtenção dos resultados, a janela de temporadas escolhida para o treinamento dos modelos foi da temporada 2008 até a temporada de 2017. Para testar os modelos gerados a partir dessa janela, foi selecionada a temporada de 2018. Dessa forma, podemos utilizar os modelos gerados a partir da janela definida anteriormente em um conjunto de dados fresco. Abaixo estão as acurácias dos modelos na janela de teste.

| | ANN | Random Forest | Kernel SVM | Naive Bayes | LSTM | Regressão Logística | Odds Baseline | Confrontos Baseline |
|---|---|---|---|---|---|---|---|---|
| **2018 - 2019** | 60.30% | 67.89% | 68.27% | 65.61% | 66.24% | 67.89% | 68.14% | 57.4% |

*Figura 20 - Resultados para Modelos de Classificação*

Como podemos ver nos resultados acima, os modelos de Random Forest, Kernel SVM e Regressão Logística foram os que mais se sobressaíram, apresentando as melhores acurácias, assim como a baseline de odds. O nosso modelo com o melhor resultado, o modelo Kernel SVM, mostrou uma alta performance pois ele não é tão influenciado por outliers, dessa forma o deixando menos sensível ao overfitting. Em contrapartida ao modelo de Kernel SVM, o modelo de Naive Bayes mostrou resultados abaixo da média, com uma acurácia de apenas 65.61%. A explicação para esse fato está nos princípios básicos desse modelo. O Naive Bayes parte da preposição de que todos os atributos possuem a mesma relevância estatística, o que não é o caso no nosso conjunto de dados, como será evidenciado mais à frente na análise de atributos

Ao analisar a performance da rede neural ANN, podemos ver que ela mostrou uma performance bem fraca na temporada de teste. Tanto ela quanto o modelo de LSTM, mostraram um overfitting nos dados de treinamento, não conseguindo se adaptar aos dados oriundos de uma nova temporada.

Como foi mencionado na seção teórica do modelo LSTM, o formato de entrada dos dados é diferente dos demais modelos. Para o modelo LSTM especificamente, o

formato de entrada é um array de três dimensões onde a primeira representa o número de instâncias, a segunda representa o número de timesteps e a terceira o número de features de entrada. O número de timesteps, em nosso contexto, representa o número de partidas anteriores que participam do conjunto de entrada. Para o caso deste modelo, o número de timesteps escolhido foi 10. Dessa forma, o modelo de LSTM começou a prever as partidas algumas iterações após o início dos outros modelos.

Ao analisar os atributos utilizados no modelo LSTM, talvez o que impeça o modelo de obter uma maior acurácia é o tratamento como uma série temporal, limitando o detalhamento mais profundo dos times e seus adversários. Os atributos de entrada do modelo são os seguintes: pontos, pontos concedidos, percentagem de lances livres, percentagem dos arremessos de quadra, percentagem de arremessos do perímetro, rebotes, turnovers, bloqueios, percentagem de vitórias na temporada até a partida em questão, percentagem de vitórias dentro de casa/como visitante, ELO, ELO do oponente, streak, PER, cotações e cotações do oponente.

### 4.3.1. Análise dos Atributos

Além de obter a acurácia atingida pelos modelos, também conseguimos gerar uma análise dos atributos a partir dos modelos desenvolvidos. Como foi dito na seção dos modelos de classificação, o modelo de random forest é capaz de fornecer a pontuação de relevância de cada atributo para o seu classificador. Esses valores, que medem a pertinência dos atributos no modelo, são computados a partir da média e desvio padrão da acumulação de diminuição de impureza em cada árvore [27]. A impureza de uma árvore é a probabilidade de que uma instância tenha o seu rótulo incorretamente previsto quando definido a partir da distribuição das instâncias no nó [28]. Abaixo, nós vemos o gráfico que mostra a importância dos atributos do modelo de random forest. Nesse gráfico, o eixo x é representado pela pontuação de relevância dos atributos, enquanto o eixo y é representado pelos atributos em si, ordenados de forma decrescente de acordo com sua pontuação.

Ao analisar o gráfico, fica evidente o impacto predominante causado pelas cotações (ODDS_A/ODDS_B), obtidas no começo do projeto quando fizemos o scrapping do site Odds Portal, seguidas dos coeficientes de ELO (ELO_A/ELO_B) das equipes e o aproveitamento na temporada (SEASON_A_PCT/SEASON_B_PCT).

Por outro lado, atributos que em teoria pareciam que funcionariam muito bem, acabaram tendo um papel minoritário nos classificadores. Alguns exemplos desses atributos são o streak de uma equipe (STREAK_A/STREAK_B), que representa a forma dela nas últimas partidas, e o histórico no confronto com aquele adversário (MATCHUP_A/MATCHUP_B). Curiosamente, a baseline de confrontos se mostrou um tanto pertinente, apresentando uma acurácia de 55% - 60% nas janelas testadas, o que bota esse resultado em perspectiva dada a pouca relevância da feature no classificador.

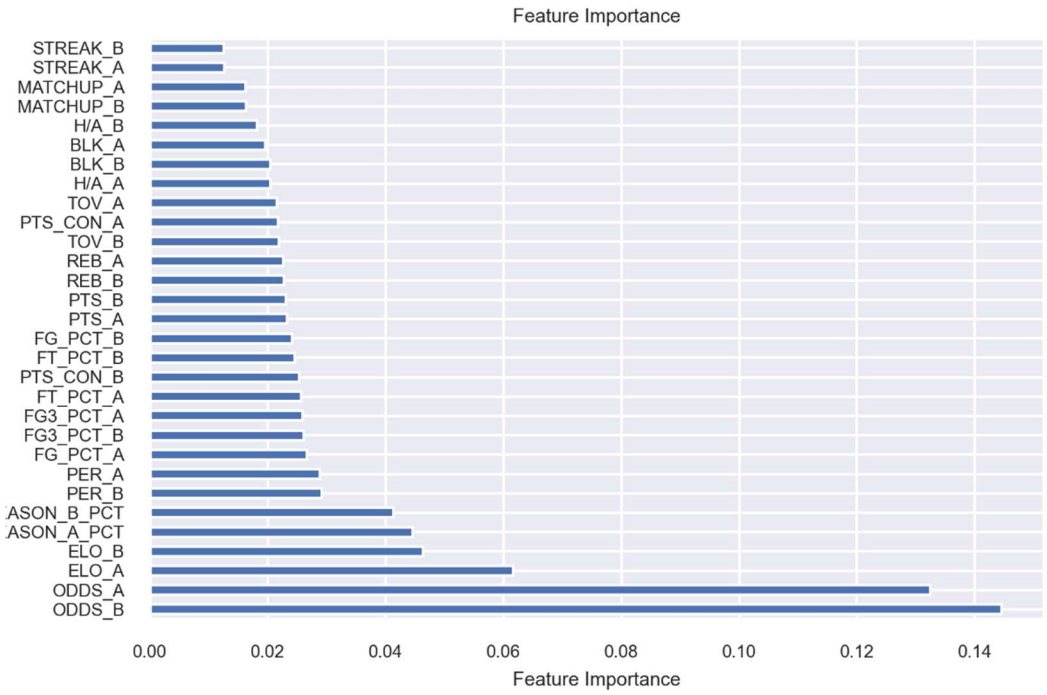



*Figura 21 - Relevância dos Atributos*

Além do gráfico de relevância de atributos, também foi gerado o gráfico de correlação de features. Esse gráfico, diferentemente da relevância de atributos, independe dos modelos gerados, visto que ele procura apenas mensurar a correlação dos atributos em nosso conjunto de dados. Apesar de o conjunto final incluir atributos referentes aos dois times, a fim de evitar redundâncias foram representados somente os atributos da equipe mandante no gráfico abaixo. Na figura 21, podemos ver a forte correlação entre dois dos atributos mais relevantes, o aproveitamento da equipe na temporada (SEASON_A_PCT/SEASON_B_PCT) e o ELO (ELO_A/ELO_B). Além disso, é interessante destacar a correlação negativa entre as cotações (ODDS_A/ODDS_B) e os atributos que representam a força da equipe, como o ELO e o PER (PER_A/PER_B). Como uma odd baixa representa que um time tem grandes

chances de ganhar aquele confronto, é muito compreensível que um valor alto de força da equipe esteja normalmente atrelado a uma baixa odd.

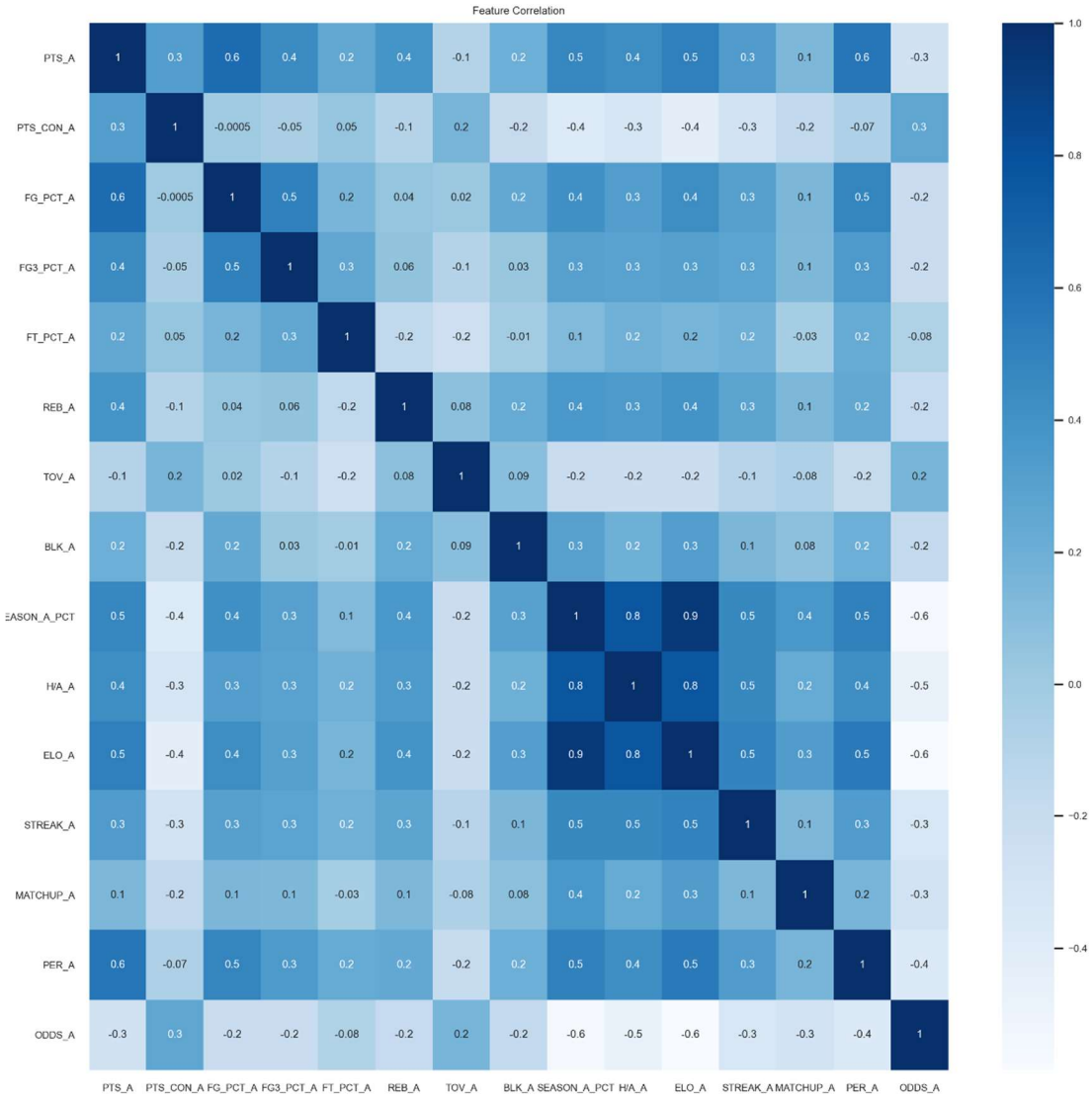



*Figura 22 - Correlação dos Atributos*

### 4.3.2. Análise Financeira

Agora vamos analisar a aplicação dos modelos considerados em relação ao retorno financeiro que eles obteriam se usados para apostar diante das cotações históricas da Odds Portal. Para atingir esse objetivo, foi feito um acompanhamento paralelo de cada modelo e retorno de cada aposta, acumulando o lucro/prejuízo obtido a cada data.

Dado um modelo de predição, utilizamos sua predição de vitória do time da casa/visitante para realizar apostas da seguinte forma: Para realizar o teste, usamos

um limiar mínimo de aposta de 1.75, ou seja, o programa nunca irá entrar em apostas cujo retorno seja menos do que 1.75 vezes o valor apostado. A quantidade a ser apostada foi definida por uma simples estratégia baseada na probabilidade de vitória do time vencedor, mais especificamente definido por: probabilidade de vitória * 10. Dessa forma, se a probabilidade do time previsto para ganhar a partida for de 60%, o nosso valor apostado será 60% de 10, ou 6. Cada modelo desenvolvido usa as suas respectivas probabilidades geradas, enquanto a probabilidade utilizada pelas baselines é a obtida pelo modelo com o melhor resultado no conjunto de teste, no caso o Kernel SVM.

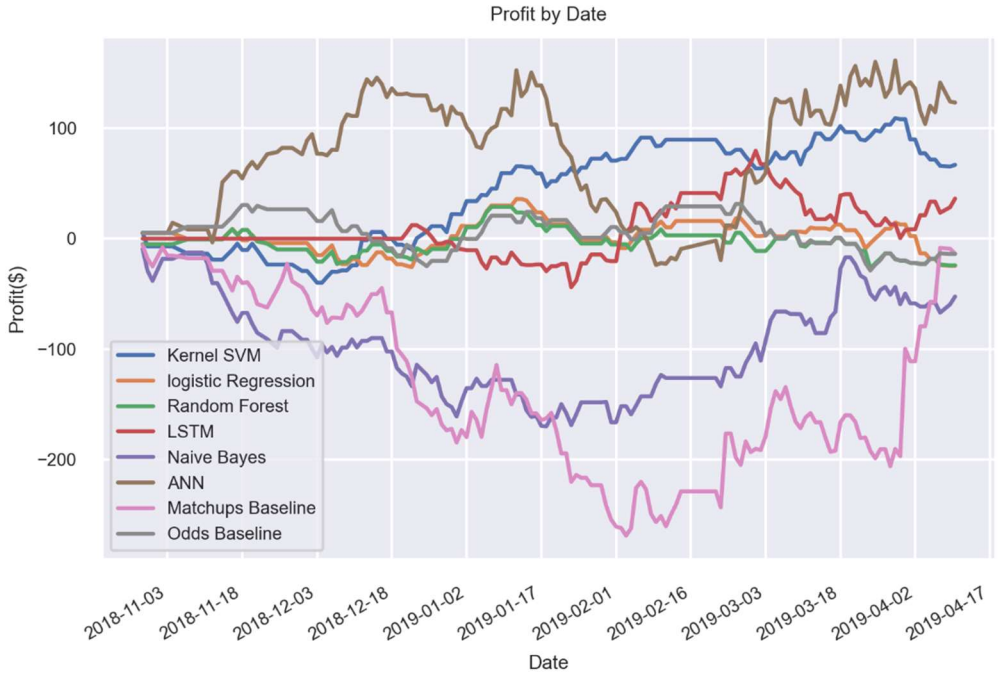



*Figura 23 - Acompanhamento Financeiro dos Modelos*

Na Figura 23, conseguimos ver o retorno financeiro de cada modelo ao longo da temporada. O eixo X representa as datas dos jogos ao longo da temporada de 2018-2019, enquanto o eixo Y representa o acompanhamento do retorno financeiro ao longo dessas datas.

Como podemos ver, os modelos de regressão logística, random forest e a baseline de odds obtiveram uma performance negativa, terminando com um retorno que varia de -50 até 0. Um destaque negativo foi o modelo naive bayes, que no contexto geral teve a pior performance e terminou com o pior retorno dentre os modelos e baselines estudados.

Antes dos destaques positivos, é importante ressaltar a recuperação da baseline de confrontos, que durante a maior parte da temporada teve o pior retorno, mas que na reta final conseguiu uma excelente performance chegando ao quarto melhor retorno com um valor um pouco abaixo de 0. Em relação aos modelos com retornos positivos, temos de exaltar o modelo ANN, que apesar de apresentar uma grande volatilidade, foi o modelo que obteve o maior lucro no final, ultrapassando a marca de 100. Um ponto interessante a se destacar na performance da ANN é que ela teve uma primeira metade de temporada muito boa, enquanto os demais modelos apresentaram resultados negativos ou não muito expressivos. Ainda nas redes neurais, a LSTM obteve uma performance muito parecida com os modelos e baseline que terminaram com prejuízo no final. A diferença é que ela conseguiu diversas apostas positivas na segunda metade da temporada, atingindo um pequeno lucro ao final do período de teste. Por fim, temos o modelo de kernel svm. O modelo que obteve a melhor acurácia do projeto, em um ponto chegou a ter um retorno de quase 100, finalizando a temporada em um lucro acima de 50, mais especificamente em 66.8.

#### 4.3.2.1. Análise do Modelo Kernel SVM

Agora vamos analisar mais afundo os resultados do nosso melhor modelo, o kernel support vector machine. Primeiramente vamos analisar as cotações das apostas acertadas e erradas. Podemos ver na Figura 24 as cotações das apostas corretas, e na Figura 25 as cotações das apostas erradas. O eixo X representa as cotações das partidas enquanto o eixo Y representa a quantidade de vezes que aquela odd esteve presente para o contexto do gráfico (aposta correta ou aposta errada). Como temos um limiar relativamente alto, podemos ver nos gráficos abaixo que tanto as apostas certas quanto erradas apresentam a faixa de 1.75 – 1.9 como as cotações mais comuns nas duas ocasiões.

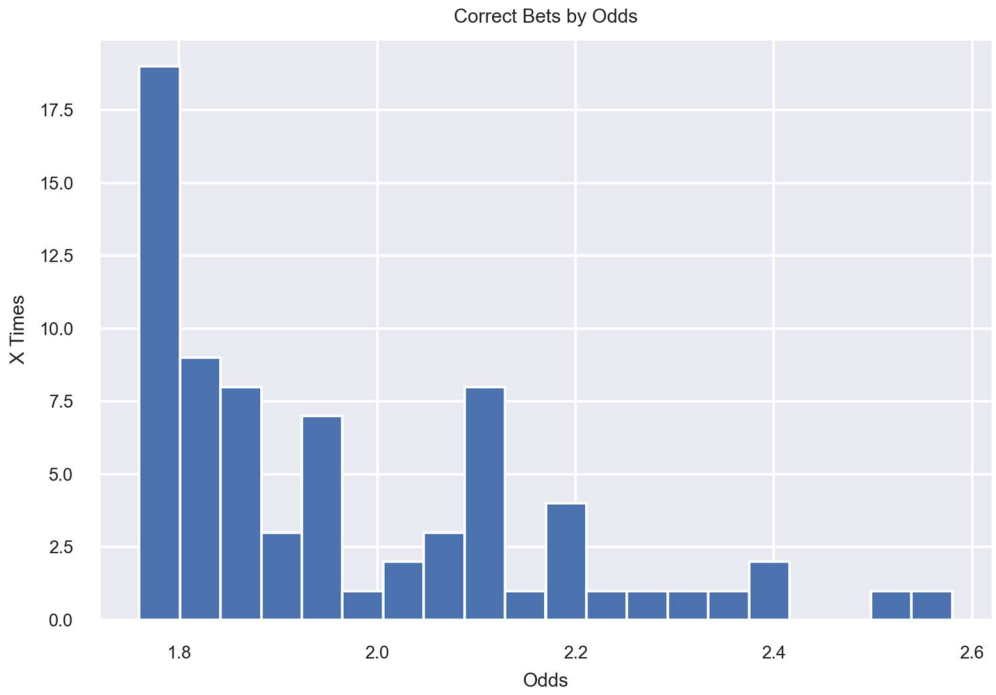



*Figura 24 - Apostas Corretas por Cotações*

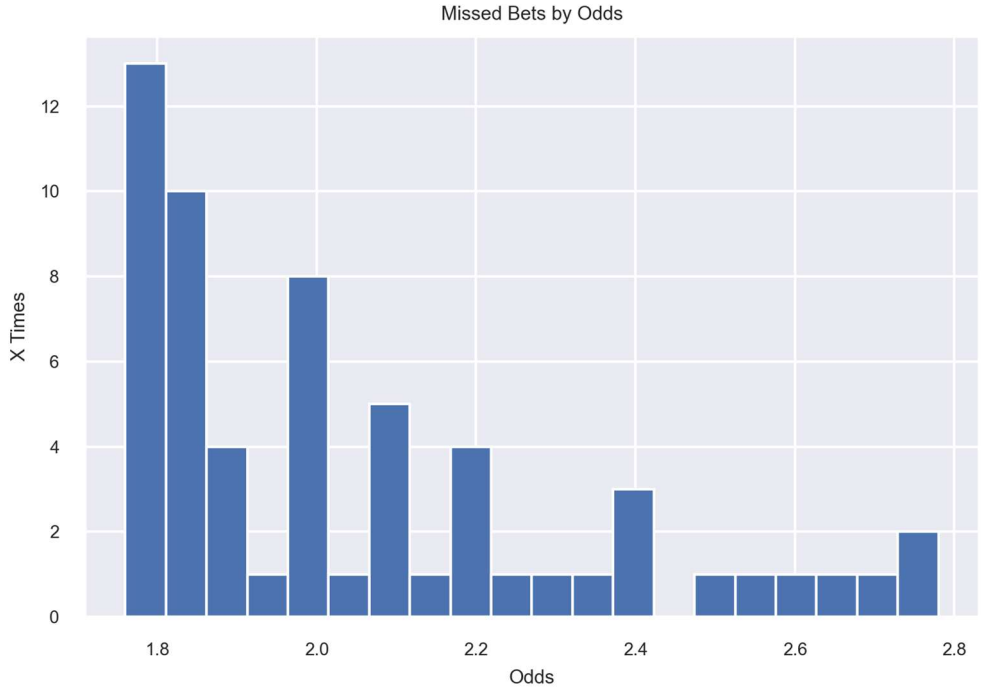



*Figura 25 - Apostas Erradas por Cotações*

Olhando por outro ponto de vista, temos a perspectiva de apostas em mandantes e visitantes. Abaixo, podemos ver na primeira imagem a parcela de apostas erradas divididas por mandante e visitante, e na imagem seguinte temos a mesma parcela só que para as apostas corretas. Como temos um limiar alto, a grande

maioria dos times que possui uma odd maior que esse threshold será mandante por conta de o time da casa geralmente ser o favorito em uma partida, o que explica a parcela majoritária dos dois gráficos sendo essas equipes (81.36% e 75.34%, respectivamente).

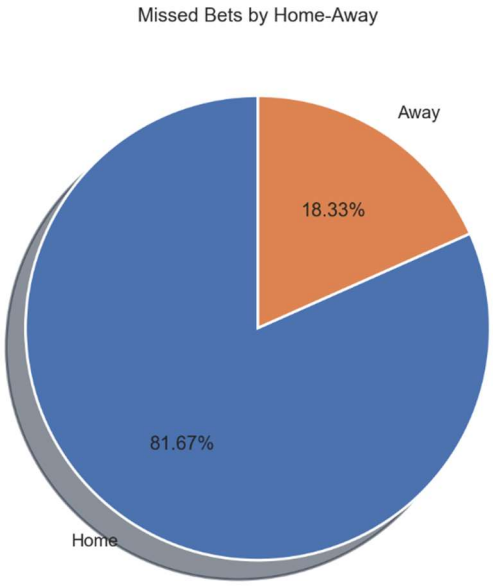



*Figura 26 - Apostas Erradas por Mandante/Visitante*

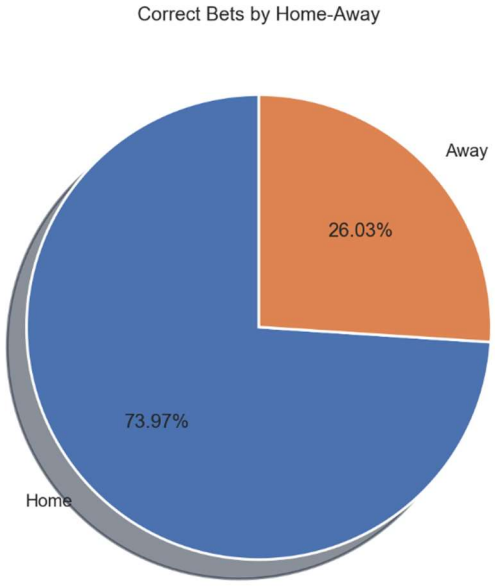



*Figura 27 - Apostas Corretas por Mandante/Visitante*

Por fim, temos na Figura 33 um gráfico que representa o retorno obtido por equipe. Para analisarmos melhor esse gráfico, é importante também analisar a tabela da NBA para a temporada 2018-19. A Figura 32 representa a classificação final da

conferência leste (à esquerda) e da conferência oeste (à direita). Nessa tabela vemos os times com o maior número de vitórias nas primeiras colocações, enquanto os times que venceram menos estão nas últimas posições.

| Conferência Leste | Conferência Oeste |
|---|---|
| 1. Milwaukee Bucks | 1. Golden State Warriors |
| 2. Toronto Raptors | 2. Denver Nuggets |
| 3. Philapelphia 76ers | 3. Portland Trail Blazers |
| 4. Boston Celtics | 4. Houston Rockets |
| 5. Indiana Pacers | 5. Utah Jazz |
| 6. Brooklyn Nets | 6. Oklahoma City Thunder |
| 7. Orlando Magic | 7. San Antonio Spurs |
| 8. Detroit Pistons | 8. Los Angeles Clippers |
| 9. Charlotte Hornets | 9. Sacramento Kings |
| 10. Miami Heat | 10. Los Angeles Lakers |
| 11. Washington Wizards | 11. Minnesota Timberwolves |
| 12. Atlanta Hawks | 12. Memphis Grizzlies |
| 13. Chicago Bulls | 13. New Orleans Pelicans |
| 14. Cleaveland Cavaliers | 14. Dallas Mavericks |
| 15. New York Knicks | 15. Phoenix Suns |

*Figura 28 - Classificação NBA 2018-2019*

Agora que temos uma perspectiva de como as equipes desempenharam esportivamente ao longo da temporada, vamos analisar o retorno obtido por cada equipe ao final da temporada. Podemos ver que é difícil encontrar um padrão. Conseguimos ver que os times que acabaram no meio da tabela ou em posições inferiores, possuem retornos que claramente se contrastam. Temos por exemplo o Miami Heat, que ficou na décima posição na tabela da conferência leste e obteve o pior retorno dentre os 30 times da liga. Já no caso do Sacramento Kings, que ficou na nona posição da conferência oeste, obteve o melhor retorno da liga, chegando a um total que ultrapassa a marca de 30. Outro ponto que vale ser destacado é o retorno obtido com os líderes das conferências, o Milwaukee Bucks (Leste) e Golden State Warriors (Oeste), que obtiveram um retorno negativo. A análise por trás disso é que como eles são os líderes, performando muito bem ao longo da temporada, eles

evidentemente vão ser os escolhidos pelo modelo como vencedores de praticamente qualquer partida a ser prevista, deixando-os muito sujeitos a vitórias inesperadas dos chamados “underdogs”.

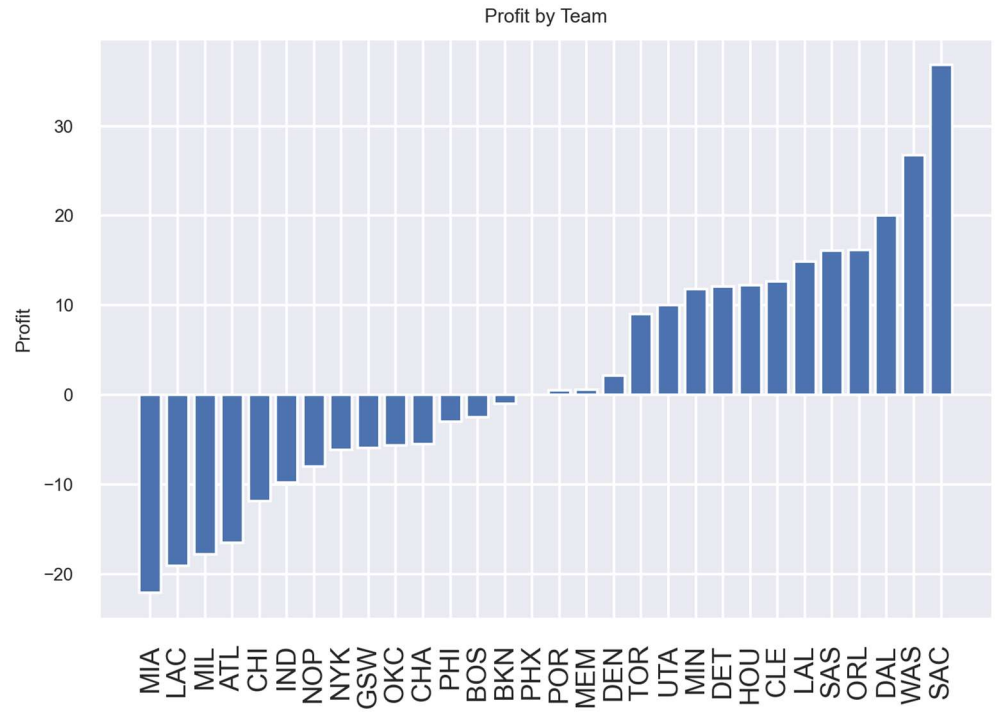



*Figura 29 - Acompanhamento Financeiro por Time*

## 5. Considerações Finais

A principal contribuição desse trabalho é um extenso acompanhamento do processo de colhimento de dados, seleção de atributos, criação de modelos e análise dos resultados em um contexto de NBA. Embora haja muitos trabalhos que contenham resultados às vezes até inacreditáveis, a maioria deles não disponibiliza os métodos descritos para alcançá-lo. Nas poucas ocasiões em que o código é disponibilizado, os resultados não são tão expressivos e às vezes não possuem uma extensa análise do que foi gerado.

O que mais me agregou ao longo da execução desse trabalho foi, além de um maior conhecimento nas técnicas e ferramentas de análise de dados e ciência de dados, foi um olhar mais estatístico para o esporte, no qual me faz olhar para certas

situações não só de basquete, mas também de outros esportes, como uma oportunidade de encontrar padrões e evidenciá-los.

Caso tivesse que começar o projeto de novo, tentaria focar em implementar menos funcionalidades, buscando melhorar as áreas mais críticas do projeto, como o aperfeiçoamento dos modelos, por exemplo. Além disso, acho que buscaria melhorar o sistema de aplicação das cotações e focaria em oferecer uma análise dos resultados mais completa. Talvez uma falha no planejamento foi tentar abranger um número considerável de modelos e funcionalidades, assim deixando de me aprofundar em um conjunto menor, podendo alcançar resultados mais expressivos.

Uma oportunidade de trabalho futuro seria a predição não somente de um vencedor, mas das estatísticas que envolvem um jogo da NBA. A partir desse objetivo base, poderíamos estender a pesquisa sobre o mercado de apostas para prever o número de turnovers, bloqueios dentre outras dúzias de estatísticas que talvez pudessem gerar um retorno maior que uma simples predição de vencedor. Ainda no mérito do mercado de apostas, também poderiam ser estudadas técnicas que otimizassem o retorno obtido.

# 6. Referências Bibliográficas


1. Patel, P. **Why Python is the most popular language used for Machine Learning.** Disponível em: https://medium.com/@UdacityINDIA/why-use-python-for-machine-learninge4b0b4457a77, Março de 2018
2. Swar. **NBA API**. Disponível em: https://github.com/swar/nba_api, Agosto de 2020
3. So, Q. **Client Case Study: Applying Machine Learning to NBA Predictions**. Disponível em: https://blog.oursky.com/2019/11/26/machine-learning-applications-nba-predictions/, Novembro de 2019
4. Johnson, W. **Machine Learning Sports Betting on the NBA Season (Before the Bubble).** Disponível em: https://medium.com/swlh/machine-learning-sports-betting-on-the-nba-season-beforethe-bubble-bd6509be7e35, Setembro de 2020
5. Silver, N. Fischer-Baum, R., **How We Calculate NBA ELO Ratings**. Disponível em: https://fivethirtyeight.com/features/how-we-calculate-nba-elo-ratings/, Março de 2015

6. Weiner, J. **Predicting the outcome of NBA games with Machine Learning.** Disponível em: https://towardsdatascience.com/predicting-the-outcome-of-nba-games-with-machinelearning-a810bb768f20, Janeiro de 2021
7. Vaidya, C. **Which NBA Statistics Actually Translate to Wins?** Disponível em: https://watchstadium.com/which-nba-statistics-actually-translate-to-wins-07-13- 2019/, Julho de 2019
8. Eremenko, K.; de Ponteves, H.; SuperDataScience Support; Ligency Team. Disponível em**: Machine Learning A-Z: Hands-On Python & R In Data Science.** https://www.udemy.com/course/machinelearning/, Última atualização Abril de 2021
9. Devlin, J. **Calculating Streaks in Pandas.** Disponível em: https://joshdevlin.com/blog/calculate-streaks-in-pandas/
10. Fein, Z. **Cracking the Code: How to Calculate Hollinger's PER Without All the Mess.** Disponível em: https://bleacherreport.com/articles/113144-cracking-the-code-how-to-calculate-hollingers-per-without-all-the-mess, Janeiro de 2009
11. Silver, N. **FiveThirtyEight**. Disponível em: https://fivethirtyeight.com/
12. MC, C. **Machine learning fundamentals (I): Cost functions and gradient descent.** Disponível em: https://towardsdatascience.com/machine-learning-fundamentals-via-linear-regression-41a5d11f5220
13. Swaminathan, S. **Logistic Regression – Detailed Overview**. Disponível em: https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc
14. Luo, S. **Loss Function (Part II): Logistic Regression**. Disponível em: https://towardsdatascience.com/optimization-loss-function-under-the-hood-part-ii-d20a239cde11
15. Chauhan, G. **All about Naïve Bayes**. Disponível em: https://towardsdatascience.com/all-about-naive-bayes-8e13cef0
16. Gandhi, R. **Understanding Hinge Loss and the SVM Cost Function**. Disponível em: https://programmathically.com/understanding-hinge-loss-and-the-svm-cost-function/
17. Navlani, A. **Decision Tree Classification**. Disponível em: https://www.datacamp.com/community/tutorials/decision-tree-classification-python

18. Bento, C. **Decision Tree Classifier explained in real-life: picking a vacation destination**. Disponível em: https://towardsdatascience.com/decision-tree-classifier-explained-in-real-life-picking-a-vacation-destination-6226b2b60575
19. IBM Cloud Education. **Neural Networks**. Disponível em: https://www.ibm.com/cloud/learn/neural-networks
20. Al-Masri, A. **How does Back-Propagation in Artificial Neural Networks Work?.** Disponível em: https://towardsdatascience.com/how-does-back-propagation-in-artificial-neural-networks-work-c7cad873ea7
21. Colah's Blog. **Understanding LSTM Networks**. Disponível em: http://colah.github.io/posts/2015-08-Understanding-LSTMs/
22. Odds Portal. **NBA Results & Historical Odds**. Disponível em: https://www.oddsportal.com/basketball/usa/nba-2020-2021/results/
23. Budhiraja, A. **Dropout in (Deep) Machine Learning**. Disponível em: https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5
24. Kurama, V. **Regression in Machine Learning: What it is and Examples of Different Models**. Disponível em: https://builtin.com/data-science/regression-machine-learning
25. Aggarwal, S. **Multiple Linear Regression**. Disponível em: https://towardsdatascience.com/multiple-linear-regression-8cf3bee21d8b
26. Scikit Learn. **R2 Score**. Disponível em: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html
27. Scikit Learn. **Feature importances with a forest of trees**. Disponível em: https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html
28. Koehrsen, W. **An Implementation and Explanation of the Random Forest in Python**. Disponível em: https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76
29. FiveThirtyEight. **The Complete History of the NBA**. Disponível em: https://projects.fivethirtyeight.com/complete-history-of-the-nba/#bucks

30. Abhigyan. **Understanding Polynomial Regression.** Disponível em: https://medium.com/analytics-vidhya/understanding-polynomial-regression-5ac25b970e18
31. Pupale, R. Support Vector Machines (SVM) – An Overview. Disponível em: https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989
32. Jain, A. Support Vector Machines (S.V.M) – Classifiers and Kernels. Disponível em: https://medium.com/@apurvjain37/support-vector-machine-s-v-m-classifiers-and-kernels-9e13176c9396
33. Scikit Learn. Randomized Search CV. Disponível em: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html
34. Brownlee, J. **A Gentle Introduction to the Rectified Linear Unit (ReLu).** Disponível em: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks
35. Malik, U. **Time Series Analysis with LSTM using Python's Keras Library**. Disponível em: https://stackabuse.com/time-series-analysis-with-lstm-using-pythons-keras-library/