



**UNIVERSIDADE FEDERAL DA FRONTEIRA SUL
CAMPUS DE CHAPECÓ
CURSO DE CIÊNCIA DA COMPUTAÇÃO**

JEFFERSON ALEXANDRE COPPINI

**USANDO APRENDIZAGEM DE MÁQUINA NA CRIAÇÃO DE MODELOS PARA
PREDIZER RESULTADOS DA LIGA NACIONAL DE FUTSAL DO BRASIL**

**CHAPECÓ
2019**

JEFFERSON ALEXANDRE COPPINI

**USANDO APRENDIZAGEM DE MÁQUINA NA CRIAÇÃO DE MODELOS PARA
PREDIZER RESULTADOS DA LIGA NACIONAL DE FUTSAL DO BRASIL**

Trabalho de conclusão de curso apresentado como requisito para obtenção do grau de Bacharel em Ciência da Computação da Universidade Federal da Fronteira Sul.
Orientador: Dr. Denio Duarte

CHAPECÓ
2019

Coppini, Jefferson Alexandre

Usando aprendizagem de máquina na criação de modelos para prever resultados da Liga Nacional de Futsal do Brasil / Jefferson Alexandre Coppini. – 2019.

63 f.: il.

Orientador: Dr. Denio Duarte.

Trabalho de conclusão de curso (graduação) – Universidade Federal da Fronteira Sul, curso de Ciência da Computação, Chapecó, SC, 2019.

1. Aprendizado de máquina. 2. Futsal. 3. Predição. 4. Modelo. 5. Algoritmos. I. Duarte, Dr. Denio, orientador. II. Universidade Federal da Fronteira Sul. III. Título.

JEFFERSON ALEXANDRE COPPINI

**USANDO APRENDIZAGEM DE MÁQUINA NA CRIAÇÃO DE MODELOS PARA
PREDIZER RESULTADOS DA LIGA NACIONAL DE FUTSAL DO BRASIL**

Trabalho de conclusão de curso apresentado como requisito para obtenção do grau de Bacharel em Ciência da Computação da Universidade Federal da Fronteira Sul.

Orientador: Dr. Denio Duarte

Este trabalho de conclusão de curso foi defendido e aprovado pela banca avaliadora em:
03/07/2019.

BANCA AVALIADORA



Dr. Denio Duarte – UFFS



Dr. Guilherme Dal Bianco – UFFS



Ma. Andressa Sebben – UFFS

RESUMO

Com base nos grandes avanços tecnológicos atuais, o volume de informações em forma de dados se torna cada vez maior. Atrelada a isso, a aprendizagem de máquina ganhou relevância, ajudando muitas áreas a encontrar padrões em grandes volumes de dados para auxiliar em problemas específicos. Uma dessas áreas, o esporte, vem utilizando técnicas de aprendizado para prever resultados de partidas, otimizar desempenho e prever situações de jogo. Neste trabalho em especial, serão apresentadas as etapas para criação de dez modelos de predição, baseados em técnicas e conceitos de aprendizado de máquina, onde tais modelos têm por objetivo prever resultados de partidas da Liga Nacional de Futsal do Brasil, com base em dados do primeiro tempo e fatores históricos da equipe. Durante a etapa de modelagem foram criadas *features* baseadas em fatores históricos que melhoraram o desempenho do modelo. Os experimentos realizados mostraram bons resultados em relação à acurácia individual, principalmente os algoritmos *Random Forest* e *Gradient Boosting*, os quais tiveram maior destaque segundo os experimentos realizados. Ainda como resultado deste trabalho, foram analisados os resultados de comitês de modelos em determinados cenários, o que caracterizou significativa melhora em relação à análise individual dos algoritmos, principalmente para previsões na classe mandante, concretizando assim a contribuição deste trabalho.

Palavras-chave: Aprendizado de máquina. Futsal. Predição. Modelo. Algoritmos.

ABSTRACT

Based on the tremendous technological advances, the data volume has increased accordingly. Machine learning has gained relevance, helping many areas find patterns in large volumes of data. Sport is one of this area. Researches are using learning techniques to predict match results and predict game situations. In this paper, we will present the steps to create ten models, based on techniques and concepts of machine learning, to predict the results of matches in the National Futsal League of Brazil. The models use data from the first half of a match and historical factors of the team. During the modeling phase, features based on historical factors were created that improved the performance of the model. The experiments performed show the models have good results concerning the accuracy, especially Random Forest and Gradient Boosting. Another contribution of this work is the committees of models. We use the committees to improve the overall accuracy, and we get better results about the individual performances. The predictions for the winning of the local team shows that the committees are essential in the prediction phase.

Keywords: Machine Learning. Futsal. Predict. Model. Algorithm.

LISTA DE ABREVIATURAS

<i>AB</i>	<i>Ada Boost</i>
<i>ACM</i>	<i>Associação Cristã de Moços</i>
<i>AM</i>	<i>Aprendizado de Máquina</i>
<i>BLC</i>	<i>Campeonato Brasileiro</i>
<i>DT</i>	<i>Decision Tree</i>
<i>EPL</i>	<i>English Premier League</i>
<i>FIFA</i>	<i>Fédération Internationale de Football Association</i>
<i>GB</i>	<i>Gradient Boosting</i>
<i>GNB</i>	<i>Gaussian Naive Bayes</i>
<i>KN</i>	<i>k-Neighbors</i>
<i>LLPD</i>	<i>La Liga Primera Division</i>
<i>LNF</i>	<i>Liga Nacional de Futsal do Brasil</i>
<i>LSVC</i>	<i>Linear Support Vector Classification</i>
<i>MLP</i>	<i>Multilayer Perceptron</i>
<i>NB</i>	<i>Naive Bayes</i>
<i>NUSVC</i>	<i>Nu Support Vector Classification</i>
<i>PCA</i>	<i>Principal Component Analysis</i>
<i>POL</i>	<i>Classificador Polinomial</i>
<i>RBF</i>	<i>Radial Base Function</i>
<i>RF</i>	<i>Random Forest</i>
<i>SVC</i>	<i>Support Vector Classification</i>
<i>SVM</i>	<i>Support Vector Machine</i>

LISTA DE ILUSTRAÇÕES

Figura 1 – Fase de classificação	15
Figura 2 – Fases finais	16
Figura 3 – Processo de aprendizado de máquina	17
Figura 4 – Esquema entre componentes do aprendizado de máquina	17
Figura 5 – Função sigmoide	19
Figura 6 – Exemplo de árvore de decisão	20
Figura 7 – Algoritmo <i>Ada Boost</i>	21
Figura 8 – Exemplo de <i>1-NN</i>	22
Figura 9 – Exemplo de rede neural	23
Figura 10 – Exemplo de hiperplanos de uma SVM	25
Figura 11 – Retornos a partir de diferentes estratégias de apostas	31
Figura 12 – Conexão entre componentes	33
Figura 13 – Diferença de RPS cumulativa	34
Figura 14 – Lucro/perda acumulado dado o fS	35
Figura 15 – Precisão média para classificação com técnica de <i>cross validation</i>	37
Figura 16 – Precisão média para classificação com técnica de <i>sliding window</i>	37
Figura 17 – Estrutura do arquivo <i>JSON</i>	40
Figura 18 – Acurácia dos modelos	50
Figura 19 – Acurácia da maioria	51
Figura 20 – Acurácia todos	51

LISTA DE TABELAS

Tabela 1 – Qualificação das <i>Features</i> pelo método <i>SelectKBest</i>	42
Tabela 2 – Média de acertos (%) em relação a melhores atributos e algoritmos	43
Tabela 3 – Relação de <i>Features</i> selecionadas	43
Tabela 4 – Média de acertos (%) com inclusão das <i>features</i> criadas	45
Tabela 5 – <i>Features</i> selecionadas para criação do modelo	46
Tabela 6 – Matriz de confusão	47
Tabela 7 – Relação de times do campeonato 2019	48
Tabela 8 – Relação de jogos preditos	49
Tabela 9 – Métrica <i>precision</i> (%)	52
Tabela 10 – Métrica <i>recall</i> (%)	53
Tabela 11 – Métrica <i>f1-score</i> (%)	53

SUMÁRIO

1	INTRODUÇÃO	11
2	O FUTSAL	13
2.1	REGRAS	13
2.2	A COMPETIÇÃO	15
3	APRENDIZADO DE MÁQUINA	17
3.1	APRENDIZADO NÃO SUPERVISIONADO	18
3.2	APRENDIZADO SUPERVISIONADO	18
3.2.1	Classificação	18
3.3	ALGORITMOS UTILIZADOS PARA CRIAÇÃO DOS MODELOS	19
3.3.1	<i>Decision Tree</i>	19
3.3.2	<i>Randon Forest</i>	20
3.3.3	Algoritmos de <i>Boost</i>	20
3.3.3.1	<i>Ada Boost</i>	20
3.3.3.2	<i>Gradient Boosting</i>	21
3.3.4	<i>Nearest Neighbors</i>	22
3.3.5	<i>Artificial Neural Network</i>	23
3.3.5.1	<i>Feed forward Neural Networks</i>	23
3.3.6	<i>Suport Vector Machine</i>	24
3.3.7	<i>Gaussian Naive Bayes</i>	25
3.4	CONSIDERAÇÕES IMPORTANTES	26
4	TRABALHOS RELACIONADOS	28
4.1	USANDO CLASSIFICAÇÕES ELO PARA PREVISÃO DE RESULTADOS	28
4.1.1	Classificações ELO	28
4.1.2	Metodologia	29
4.1.3	Medidas avaliativas	30
4.1.4	Principais resultados	31
4.1.5	Comentários Finais	31
4.2	UM MODELO DE REDE BAYESIANA PARA PREVISÃO DE RESULTADOS	32
4.2.1	Medição de precisão	33
4.2.2	Medição de rentabilidade	34
4.2.3	Comentários Finais	35
4.3	EXPLORANDO CLASSIFICADOR POLINOMIAL PARA PREVER RESULTADOS DE PARTIDAS	35
4.3.1	Classificador Polinomial (POL)	36
4.3.2	Metodologia	36
4.3.3	Principais resultados	37

4.3.4	Comentários Finais	38
4.4	CONSIDERAÇÕES IMPORTANTES	38
5	PROJETO DOS EXPERIMENTOS	40
5.1	OBTENÇÃO DOS DADOS	40
5.2	CRIAÇÃO DO MODELO	41
5.2.1	<i>Features</i> do Modelo	41
5.2.1.1	Seleção de <i>Features</i>	41
5.2.1.2	Criação de <i>Features</i>	43
5.2.2	Treinamento	46
5.3	AVALIAÇÃO DO MODELO	46
6	EXPERIMENTOS E RESULTADOS	48
7	CONCLUSÃO	54
	REFERÊNCIAS	55
	APÊNDICE A – PARÂMETROS	57
	APÊNDICE B – PARÂMETROS SELECIONADOS	61

1 INTRODUÇÃO

O mundo se desenvolve a cada dia em vários segmentos da sociedade. A tecnologia tem papel importante nessa evolução, seja ela por desenvolvimento de aspectos de infraestrutura, como o *hardware*, que se moderniza constantemente, ou de aspectos intelectuais, na qual grandes volumes de dados são gerados todos os dias. Com base nesses avanços, houve igualmente um aumento na capacidade de armazenar e processar grandes quantidades de dados, bem como acessá-los de locais fisicamente distantes em uma rede de computadores (ALPAYDIN, 2010).

Em torno desse grande volume de dados, técnicas de aprendizado de máquina no esporte ganharam força, especificamente na área de predição. A aplicação de métodos avançados de aprendizado de máquina a conjuntos de dados grandes (geralmente chamados de "*big data*") podem revelar novas intuições que, de outra forma, permaneceriam desconhecidos (A. CONSTANTINOU; N. FENTON, 2017). Fatores como desempenho histórico das equipes, resultados de partidas e dados sobre os jogadores, ajudam os diferentes interessados a entender as chances de ganhar ou perder partidas futuras. Portanto, o desafio de prever resultados esportivos é algo que há muito tempo interessa a diferentes públicos, sejam eles casas de apostas, torcedores, potenciais interessados e a mídia (BUNKER; THABTAH, 2017).

Há um grande número de fatores que podem afetar o resultado de um partida. Uma das dificuldades em qualquer investigação das relações envolvidas em um dado evento é que, em grande parte, a suposição de um determinado modelo determina os atributos que são estudados e predetermina as possíveis relações que podem ser encontradas. Assim, o ato de escolher qual modelo e atributos estudar limita o que pode ser descoberto (JOSEPH; N. E. FENTON; NEIL, 2006).

Ao abordar um novo problema, existem duas técnicas comumente usadas. A primeira assume que tem-se alguma ideia de como a situação sob investigação funciona, então um modelo é construído, e usando esse modelo, são selecionados os atributos que acredita-se contribuir para o evento investigado. A segunda abordagem assume pouco conhecimento dos mecanismos subjacentes envolvidos, de modo que analisa-se todos os atributos provavelmente relevantes e se tenta determinar aqueles que têm o efeito mais significativo. Isso ainda está em vigor na construção de um modelo *a priori*, mas apenas muito informal (JOSEPH; N. E. FENTON; NEIL, 2006). Este trabalho foca no princípio da primeira abordagem, ao se tratar da ideia de entendimento do cenário do jogo, para fazer previsões futuras.

No aspecto financeiro, nos últimos anos, o ramo de apostas de futebol experimentou o crescimento mais rápido nos mercados de jogos de azar. Além do mais, o mercado do futebol europeu excedeu os 25 bilhões de euros em 2016/17, enquanto o mercado global de jogos esportivos é estimado em até 3 trilhões (A. C. CONSTANTINOU, 2018; A. C. CONSTANTINOU; N. E. FENTON; NEIL, 2012). As probabilidades propostas pelas casas de apostas frequentemente são influenciadas por volumes de apostas e, portanto, nem sempre refletem a probabilidade real dos resultados da partida. De fato, um dos objetivos das casas de apostas é

encorajar os apostadores a subdividir suas apostas. Ao fazê-lo, eles minimizam o risco e ganham com as probabilidades propostas. Além disso, as casas de apostas podem definir sistematicamente as probabilidades para tirar proveito dos preconceitos dos apostadores, como a conhecida preferência por favoritos e equipes locais, para aumentar os lucros. Portanto, a comparação entre chances reais e probabilidades pode ser explorada para definir uma estratégia de apostas lucrativa (ANGELINI; DE ANGELIS, 2017).

Por sua popularidade, o futebol atrai o maior número de trabalhos relacionados a predição, porém, nem sempre os objetivos são os mesmos. Por exemplo, o trabalho, (A. C. CONSTANTINOU; N. E. FENTON; NEIL, 2012) é relacionado à predição de resultados de partidas. Já (HVATTUM; ARNTZEN, 2010) procura comparar métodos de predição, (JOSEPH; N. E. FENTON; NEIL, 2006) prevê o desenvolvimento de um determinado time, (ANGELINI; DE ANGELIS, 2017) visa calcular a probabilidade de cada resultado e (HUANG; CHANG, 2010) tem por objetivo prever o resultado final de uma competição.

Este trabalho apresenta as etapas da criação de dez modelos de predição tendo como propósito prever resultados de partidas de futsal, em particular da Liga Nacional de Futsal do Brasil (LNF). Especificamente, os modelos construídos neste trabalho são capazes de prever resultados das partidas baseado em *features* extraídas de características do primeiro tempo e *features* criadas a partir de fatores históricos da equipe na competição. Em seguida, foram selecionadas as *features* mais informativas utilizando uma técnica de seleção de melhores atributos. Estes modelos foram criados utilizando técnicas de aprendizado de máquina supervisionadas. Os resultados obtidos indicam que os modelos tiveram bom desempenho individual, além de de resultados expressivos utilizando uma proposta de comitê para melhorar a predição.

O prosseguimento deste trabalho está estruturado da seguinte forma: no Capítulo 2 é exibido o conceito de futsal, as regras são apresentadas e é explicado a estrutura do campeonato. O Capítulo 3 introduz o referencial teórico sobre aprendizado de máquina e suas técnicas. Os trabalhos relacionados utilizados como base estão no Capítulo 4. No Capítulo 5 é apresentado o projeto de experimentos, assim como no Capítulo 6 são descritos os experimentos utilizados e os resultados obtidos. Por fim o Capítulo 7, sistematiza a contribuição desse trabalho.

2 O FUTSAL

O futsal ou futebol de salão é um esporte que surgiu em meados do século XX, e assim como em outras modalidades esportivas, possui divergência quanto ao seu surgimento. Porém a tese mais predominante é que o futsal foi criado no Uruguai por volta de 1934, pelo professor de Educação Física da Associação Cristã de Moços (ACM) Juan Carlos Ceriani Gravier, nomeando o novo esporte de *Indoor Football*. A ideia teria sido baseada em aspectos do futebol, motivado pelo destaque do esporte no país, e por consequência, os resultados internacionais obtidos por sua seleção (FUTEBOL DE SALÃO - CBFS, 2018).

Atualmente a instituição reguladora do esporte é a *Fédération Internationale de Football Association* (FIFA). A mesma é responsável por organizar as competições de âmbito mundial, entre elas, a copa do mundo de futsal, realizada de quatro em quatro anos, com representantes de todos os continentes. Além de competições internacionais, o futsal está presente como liga Nacional em vários países, entre elas o Brasil, um dos objetos de estudo desse trabalho.

Este capítulo está dividido em duas seções. A primeira diz respeito a algumas regras do esporte, e como essas regras tornam algumas características do jogo importantes para criação do modelo. Na segunda seção desse capítulo será abordada a estrutura da competição e quais características podem ser juntadas ao modelo para melhor descrever o cenário de um jogo.

2.1 REGRAS

A criação de modelos de aprendizado de máquina, que será assunto no próximo capítulo, é baseada em características do conjunto de dados de entrada (também conhecidos com *features*). Nesta seção, serão apresentadas regras que podem ser utilizadas para construir *features* para compor o modelo proposto para esse trabalho, como por exemplo, número de faltas cometidas, número de faltas sofridas, cartões amarelos recebidos, cartões vermelhos sofridos, chutes, entre outros. Seguem abaixo algumas regras do esporte que podem ser importantes na hora de definir quais características representam melhor cada cenário (FUTEBOL DE SALÃO CBFS, 2018).

- **Quadra de Jogo:** os jogos deverão ser disputados em superfícies lisas. O seu piso será construído de madeira ou material sintético rigorosamente nivelado, sem declives ou depressões. A quadra de jogo terá medidas de no mínimo 38 metros de comprimento por 18 metros de largura;
- **Número de Jogadores:** a partida será disputada entre duas equipes compostas, cada uma, por no máximo 05 (cinco) jogadores, um dos quais será o goleiro. O número máximo de jogadores reservas, para substituições é de 9 (nove) jogadores, onde será permitido um número indeterminado de substituições durante a partida;

- **Duração da Partida:** o tempo de duração de uma partida será cronometrado e dividido em dois períodos de 20 minutos cada. Observação: O cronômetro é parado quando a bola sai de jogo ou quando ocorre uma falta, e só volta a correr quando a partida é reiniciada;
- **Contagem de Gols:** será válido o gol quando a bola ultrapassar inteiramente a linha de meta entre os postes de meta e sob o travessão, contanto que não tenha sido cometida nenhuma infração por jogador atacante. A equipe que tenha marcado maior número de gols será considerada vencedora da partida. Se houver igualdade no número de gols assinalados por cada equipe ou se nenhum for marcado pelas equipes disputantes, a partida será considerada empatada;
- **Faltas e Incorreções:** será concedido um tiro livre direto em favor da equipe adversária quando um jogador cometer infrações contra jogador adversário, de maneira que os árbitros julguem imprudente, temerária ou com uso de força excessiva;
 - a) Cartão Amarelo: o árbitro poderá apresentar cartão amarelo a um jogador quando ele interpretar e julgar conduta antidesportiva ou por infração recorrente às regras do jogo;
 - b) Cartão Vermelho: o árbitro poderá apresentar cartão vermelho a um jogador quando ele interpretar e julgar culpado por entrada brusca, conduta violenta, linguagem ou gestos ofensivos, impedimento de forma ilegal seu adversário de assinalar um gol e praticar pela segunda vez infração passível de cartão amarelo na mesma partida;
- **Tiros Livres:** os tiros livres podem ser divididos em duas categorias: o tiro livre indireto e o tiro livre direto. No tiro livre indireto o jogador que cobra a falta não pode fazer o gol diretamente a partir da cobrança; já no direto essa condição é possível. O tiro livre direto sem barreira é dado ao time que sofre a sexta falta no tempo de jogo. A cobrança pode ser feita a partir do local em que a falta ocorreu ou na sinalização da segunda marca penal;
- **Tiro Penal:** o tiro penal é um tipo de tiro livre que é dado ao time que sofrer alguma infração dentro da área adversária. Essa cobrança é efetuada sem barreira na sinalização da primeira marca penal.

As regras acima descritas modelam a dinâmica de um jogo de futsal, e, por consequência o resultado de uma partida. A informação de que um time cometeu um número excessivo de faltas, pode induzir a um pensamento que esse time esteve mais vulnerável durante o tempo de jogo. O mesmo modo de pensar pode ser expandido para o número de cartões amarelos, um cartão amarelo pode significar uma falta, ou em alguns casos uma chance de gol ao adversário. Uma extensão do cartão amarelo, o cartão vermelho, agrega todas as vulnerabilidades do cartão amarelo com o agravante na interferência na estrutura do time, por se tratar de uma inferioridade numérica de jogadores durante um período de tempo.

Além das características citadas acima, fatores como número de finalizações, posse de bola, goleiro linha, se adaptam a questões que afetam estruturalmente o jogo. A partir destas































informações pode se inferir o modelo tático de determinado time (defensivo ou ofensivo) e criar um padrão dos mesmos. Algumas dessas características não serão utilizadas, pois o conjunto de dados disponibilizado pela Liga Nacional de Futsal do Brasil não possui tais informações.






2.2 A COMPETIÇÃO

Assim como na seção anterior, a ideia é mostrar características que poder usadas na construção de *features* do modelo. Neste sentido, a segunda parte da criação de *features* se dará por meio das características da competição da Liga Nacional de Futsal do Brasil, objeto de estudo desse trabalho.

Na primeira fase da competição, o número de times pode variar entre dezessete a dezenove. A Figura 1 mostra a classificação final após a primeira fase da competição de 2017 composta por dezessete times. Percebe-se na coluna **J** (jogos disputados) que cada equipe jogou dezesseis vezes, ou seja todos os times jogaram contra todos apenas uma vez, onde os dezesseis melhores times da competição avançam para próxima fase.

Figura 1 – Fase de classificação

CLASSIFICAÇÃO GERAL										
Equipe	P	J	V	E	D	GP	GC	SG	%	Ult. Jogos
1  Magnus	31	16	9	4	3	65	42	23	64,6%	
2  Joinville	30	16	8	6	2	53	42	11	62,5%	
3  ACBF	28	16	8	4	4	46	33	13	58,3%	
4  Corinthians	28	16	7	7	2	37	27	10	58,3%	
5  Jaraguá	26	16	7	5	4	40	37	3	54,2%	
6  Pato	24	16	7	3	6	51	46	5	50,0%	
7  Atlântico	24	16	6	6	4	36	36	0	50,0%	
8  Assoeva	23	16	6	5	5	43	42	1	47,9%	
9  Copagril	23	16	6	5	5	36	40	-4	47,9%	
10  Joaçaba	22	16	6	4	6	33	35	-2	45,8%	
11  Foz Cataratas	22	16	6	4	6	34	41	-7	45,8%	
12  Marreco	22	16	4	10	2	37	34	3	45,8%	
13  Intelli	18	16	5	3	8	44	43	1	37,5%	
14  Minas	17	16	4	5	7	39	46	-7	35,4%	
15  Concórdia	12	16	3	3	10	29	38	-9	25,0%	
16  Tubarão	12	16	3	3	10	28	44	-16	25,0%	
17  Guarapuava	7	16	2	1	13	34	59	-25	14,6%	

 Permaneceu na mesma posição -  Subiu de posição -  Caiu de posição /  Empate -  Vitória -  Derrota
P: Pontos - **J:** Jogos - **V:** Vitórias - **E:** Empates - **D:** Derrotas - **GP:** Gols Pro - **GC:** Gols Contras - **SG:** Saldo de Gols - **%:** Aproveitamento de pontos - **Ult. Jogos** Últimos jogos.

FONTE: (FUTSAL, 2017)

Os dezesseis times classificados para a segunda fase são divididos em oito chaves, sendo a estrutura do chaveamento e os confrontos definidos pela classificação da equipe na primeira fase, conforme Figura 2. Por exemplo, o time mais bem colocado na fase de classificação (*I.E.*, Magnus) pega o décimo sexto time mais bem classificado na fase anterior (*I.E.*, Tubarão), o segundo pega o décimo quinto, e assim por diante. O classificado se dará por resultado agregado em jogos de ida e volta.

Figura 2 – Fases finais



FONTE: (FUTSAL, 2017)

Semelhante à segunda fase, na terceira, os oito classificados serão dispostos em quatro chaves com duas equipes conforme chaveamento da fase anterior. O classificado se dará por resultado agregado em jogos de ida e volta.

Na quarta fase ou fase semifinal os quatro times classificados serão dispostos em duas chaves com duas equipes conforme chaveamento da fase anterior. O classificado se dará por resultado agregado em jogos de ida e volta.

E por fim na grande final, estarão os dois times classificados nas semifinais e que disputarão o título. Assim como em fases anteriores a final é disputada em jogos de ida e volta onde o campeão se dará pelo resultado agregado das duas partidas.

A ideia em de trazer informações do campeonato para o modelo é atrelar as características do jogo citadas na seção anterior com padrões específicos para cada fase da competição, além de entender a força de determinado mandante ou visitante na fase específica e por consequência caracterizar um cenário com fatores que vão além do campo de jogo.

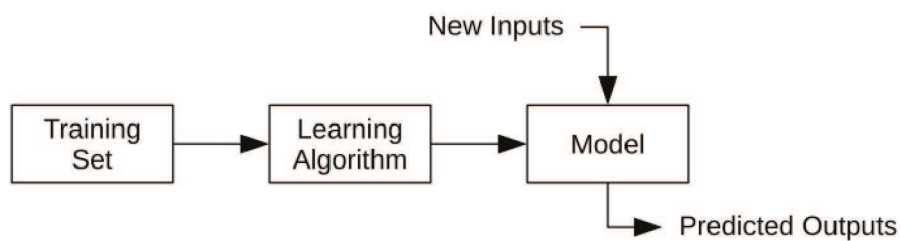
3 APRENDIZADO DE MÁQUINA

Um dos ramos da Inteligência Artificial, o aprendizado de máquina (AM), visa encontrar padrões em grandes volumes de dados. Tais padrões podem auxiliar a entender o processo, ou fazer previsões sobre o modelo, supondo que o futuro, pelo menos no futuro próximo, não seja muito diferente do passado quando os dados da amostra foram coletados (ALPAYDIN, 2010).

Os dados são a entrada de qualquer sistema de aprendizado de máquina, pois os mesmos contêm exemplos de um determinado domínio. Os dados usados como entrada no modelo de treinamento são chamados de dados de treinamento (DUARTE; STÅHL, 2019).

A Figura 3 é uma representação abstrata do processo de aprendizado de máquina, tal que o conjunto de treinamento (*Training Set*) funciona como entrada para o algoritmo de aprendizagem de máquina (*Learning Algorithm*), responsável por construir um modelo (*Model*) matemático com base em suas características. Com o modelo construído, o próximo passo é propor novos cenários (*New Inputs*) para que novas saídas sejam preditas (*Predicted Outputs*).

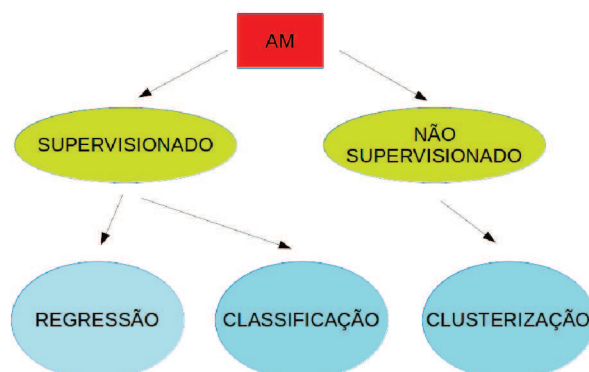
Figura 3 – Processo de aprendizado de máquina



FONTE: (DUARTE; STÅHL, 2019)

A Figura 4 esquematiza as relações entre ferramentas e técnicas que são necessárias para resolver problemas utilizando aprendizagem de máquina. No aprendizado de máquina supervisionado, utilizam-se ferramentas de classificação e regressão para construção do modelo, enquanto o aprendizado não supervisionado se restringe na maioria das vezes a ferramentas de clusterização. Os dois tipos de aprendizagem serão apresentados brevemente a seguir.

Figura 4 – Esquema entre componentes do aprendizado de máquina



3.1 APRENDIZADO NÃO SUPERVISIONADO

Na aprendizagem não supervisionada não existe mapeamento entre os exemplos e o que eles representam. Neste método de aprendizado, as entradas não possuem um rótulo e necessitam ser agrupadas para definir possíveis padrões, o que é feito através da clusterização (ALPAYDIN, 2010; SHAI SHALEV-SHWARTZ, 2013).

A aprendizagem não supervisionada é menos objetiva do que a aprendizagem supervisionada, uma vez que não há rótulos para orientar o usuário para a análise. O domínio do conjunto de dados deve ser conhecido pelo usuário para construir modelos úteis, caso contrário, os resultados podem não ser compreensíveis (DUARTE; STÅHL, 2019).

3.2 APRENDIZADO SUPERVISIONADO

Diferentemente do aprendizado não supervisionado, no aprendizado supervisionado as características de entrada possuem significado, ou seja, rótulos que as define. Neste tipo de aprendizagem, o objetivo é aprender um mapeamento da entrada para uma saída cujos valores corretos são fornecidos por um rótulo, assim atribuindo valor às características de entrada (ALPAYDIN, 2010). Regressão e classificação são as duas principais técnicas desse tipo de aprendizado. Neste trabalho, o foco se dará no método de classificação, o qual será explicado a seguir.

3.2.1 Classificação

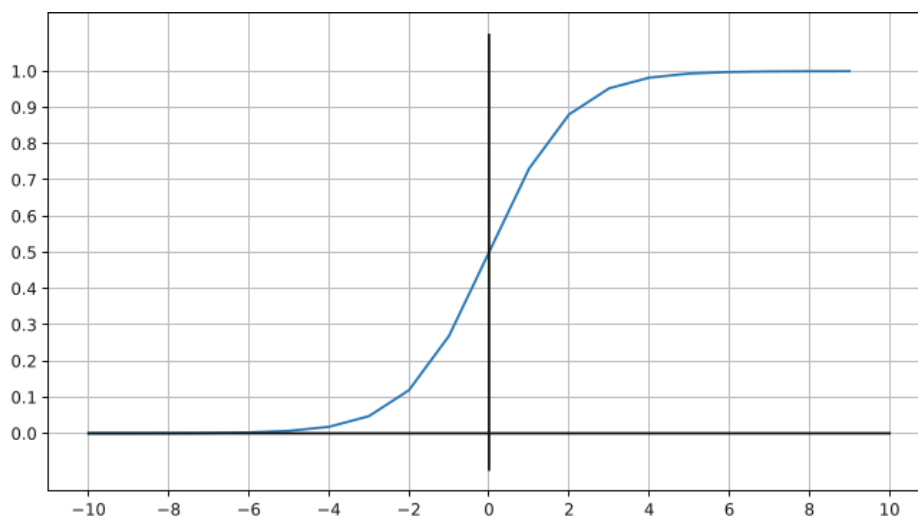
Com objetivo muito semelhante à regressão, a ferramenta de classificação é muito utilizada para situações em que se tem conhecimento dos possíveis valores para rótulos, ou seja, a classificação para sua entrada é delimitada por rótulos discretos. Esses rótulos representam classes que podem ser binária ou multi-classes.

Uma das ferramentas eficientes para esses casos é a regressão logística. A classe de hipóteses associada à regressão logística é a composição de uma função sigmoide σ , $\mathbb{R} \rightarrow [0, 1]$ (SHAI SHALEV-SHWARTZ, 2013), portanto a imagem da função sigmoide estará entre 0 e 1, o que facilita delimitar algum valor entre esse intervalo que determine a qual classe pertence o resultado. A função sigmoide pode ser definida como:

$$f_{\sigma}(z) = \frac{1}{1 + e^{-z}}$$

A Figura 5 representa uma função sigmoide, onde sua imagem sempre estará no intervalo $[0,1]$. Por sua vez, percebe-se que os limites para função são peculiares, pois seu limite com a entrada tendendo a menos infinito será zero e seu limite com entrada tendendo a mais infinito será 1, portanto os valores nunca vão estar exatamente em zero ou um. Não existe um padrão para delimitar os valores para classificação, o mais é utilizado é a metade do intervalo, porém muito depende da aplicação. Perceba que quando $z = 0$, o valor da sigmoide será 0.5.

Figura 5 – Função sigmoide



FONTE: (DUARTE; STÅHL, 2019)

3.3 ALGORITMOS UTILIZADOS PARA CRIAÇÃO DOS MODELOS

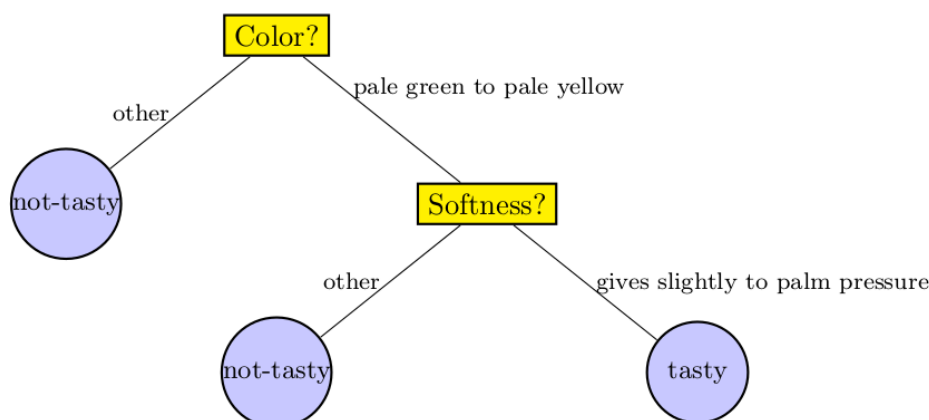
Nesta seção serão apresentados os algoritmos que implementam as funções executadas para criação dos dez modelos de predição utilizados nesse trabalho.

3.3.1 *Decision Tree*

Uma árvore de decisão é um preditor, $h: X \rightarrow Y$, que prevê o rótulo associado a uma instância x percorrendo de um nó raiz de uma árvore para uma folha (SHAI SHALEV-SHWARTZ, 2013). Ou seja, o rótulo de determinada entrada sempre estará nos nós folha, pois sua caracterização é feita a partir do deslocamento na árvore. Essa técnica pode ser usada tanto em algoritmos de classificação, quanto regressão, além de ser um dos métodos mais utilizados, baseado em sua forma estrutural de fácil visualização e entendimento.

A Figura 6 mostra um exemplo prático de como a árvore de decisão funciona, transformando as características do modelo em condicionais dos nós da árvore e os possíveis rótulos em folhas. O nó raiz da árvore possui um delimitador do problema, pois é a partir dele que se começa a caracterização. No exemplo da Figura 6, o nó raiz da árvore pergunta sobre a cor da fruta, as arestas ligadas ao nó referenciam as possibilidades, neste caso se a cor do mamão for verde pálida ou amarela pálida verifica-se a próxima condição, caso contrário o mamão não é saboroso. Visualizando o próximo nó da árvore, seu condicional é em relação a suavidade do mamão, se sua maciez proporcionar uma pressão na palma, o mamão é saboroso, caso contrário não é.

Figura 6 – Exemplo de árvore de decisão



FONTE: (SHAI SHALEV-SHWARTZ, 2013)

3.3.2 *Random Forest*

A classe de árvores de decisão em casos extremos pode ter dimensão infinita. Isso pode se tornar um problema levando em consideração o tamanho do conjunto de dados. Uma das maneiras de reduzir o perigo de *overfitting* é construir um conjunto de árvores, também conhecidas por *Random Forests* (SHAI SHALEV-SHWARTZ, 2013).

O *Random Forest* é um classificador que consiste em uma coleção de árvores de decisão, onde cada árvore é construída aplicando um algoritmo A no conjunto de treinamento S e um vetor aleatório adicional θ , onde θ é amostrado com alguma distribuição de probabilidade. O valor predito é obtido por uma votação majoritária sobre as previsões das árvores individuais (SHAI SHALEV-SHWARTZ, 2013).

3.3.3 Algoritmos de *Boost*

O *Boost* tem sido uma técnica muito bem sucedida para resolver o problema de classificação de duas ou mais classes (HASTIE et al., 2009). Algoritmos como *Ada Boost* e *Gradient Boosting* são dois dos principais algoritmos baseados nessa técnica. Ambos serão apresentados brevemente a seguir.

3.3.3.1 *Ada Boost*

O algoritmo *Ada Boost*, recebe como entrada um conjunto de treinamento chamado S , tal que $S = (x_1, y_1), \dots, (x_m, y_m)$, onde para cada i , $y_i = f(x_i)$ para alguma função que rotula f . O processo de reforço prossegue em uma sequência de rodadas consecutivas. Na rodada t ,

o *booster* define primeiro uma distribuição sobre os exemplos em S , denotada por $D^{(t)}$. Isto é $D^{(t)} \in \mathbb{R}_+^m$ e $\sum_{i=1}^m D_i^{(t)} = 1$. Então, o *booster* passa a distribuição $D^{(t)}$ e a amostra S para o preditor fraco (SHAI SHALEV-SHWARTZ, 2013). Supõe que o preditor retorna uma hipótese fraca, h_t , cujo erro:

$$e_t = L_{D^{(t)}}(h_t) = \sum_{i=1}^m D_i^{(t)} \mathbb{I}_{[h_t(x_i) \neq y_i]}$$

é no máximo $\frac{1}{2} - \gamma$. Então o *Ada Boost* atribui um peso para h_t : $w_t = \frac{1}{2} \log(\frac{1}{e_t} - 1)$. O peso de h_t é inversamente proporcional ao erro de h_t . No final da rodada, o *Ada Boost* atualiza a distribuição de modo que os exemplos em quais erros h_t obterão uma massa de probabilidade mais alta, enquanto os exemplos em que h_t está correto obterão uma massa de probabilidade mais baixa. Intuitivamente, isso forçará o preditor fraco a se concentrar nos exemplos problemáticos da próxima rodada. A saída do algoritmo *Ada Boost* é um classificador forte que é baseado em uma soma ponderada de todas as hipóteses fracas (SHAI SHALEV-SHWARTZ, 2013). O algoritmo apresentado na Figura 7 resume este processo.

Figura 7 – Algoritmo *Ada Boost*

```

AdaBoost

input:
    training set  $S = (\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)$ 
    weak learner WL
    number of rounds  $T$ 
initialize  $\mathbf{D}^{(1)} = (\frac{1}{m}, \dots, \frac{1}{m})$ .
for  $t = 1, \dots, T$ :
    invoke weak learner  $h_t = \text{WL}(\mathbf{D}^{(t)}, S)$ 
    compute  $\epsilon_t = \sum_{i=1}^m D_i^{(t)} \mathbb{I}_{[y_i \neq h_t(\mathbf{x}_i)]}$ 
    let  $w_t = \frac{1}{2} \log\left(\frac{1}{\epsilon_t} - 1\right)$ 
    update  $D_i^{(t+1)} = \frac{D_i^{(t)} \exp(-w_t y_i h_t(\mathbf{x}_i))}{\sum_{j=1}^m D_j^{(t)} \exp(-w_t y_j h_t(\mathbf{x}_j))}$  for all  $i = 1, \dots, m$ 
output the hypothesis  $h_s(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^T w_t h_t(\mathbf{x})\right)$ .

```

FONTE: (SHAI SHALEV-SHWARTZ, 2013)

3.3.3.2 Gradient Boosting

O *Gradient Boosting* é uma técnica de aprendizado de máquina que pode resolver tanto problemas de classificação quanto de regressão. Esta técnica constrói sua estrutura com base em preditores fracos, geralmente árvores de decisão. Sua construção utiliza técnica de *boost* em sua essência, utilizando a função gradiente como forma de otimização do método (FRIEDMAN, 2002).

O *Gradient Boosting* constrói modelos sequenciais definindo uma função parametrizada simples para os resíduos atuais por mínimos quadrados em cada iteração. Os resíduos são o gradiente da perda funcional sendo minimizado, com relação aos valores do modelo em cada ponto de dados de treinamento avaliado na etapa atual (FRIEDMAN, 2002).

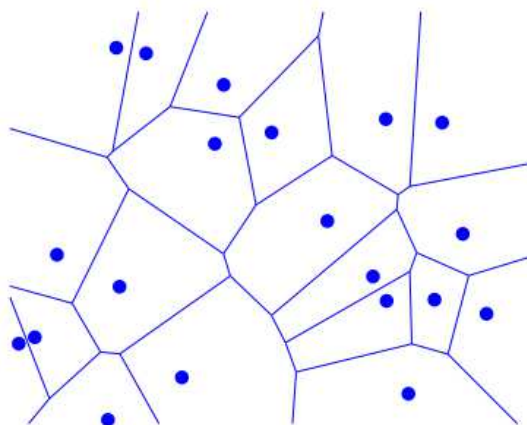
3.3.4 *Nearest Neighbors*

Algoritmo *Nearest Neighbors* (NN) ou algoritmo de vizinhos mais próximos é um outro tipo de técnica de aprendizagem de máquina. A ideia é memorizar o conjunto de treinamento e depois prever o rótulo de qualquer nova instância com base nos rótulos de seus vizinhos mais próximos no conjunto de treinamento (SHAI SHALEV-SHWARTZ, 2013). A motivação é utilizar a computação para que os vizinhos de determinado ponto tenham o mesmo rótulo que esse ponto. Depois dessa vizinhança gerada a tarefa de encontrar o rótulo para uma situação se torna mais fácil.

Dentro dessa técnica do vizinho mais próximo existe o algoritmo de *K-Nearest Neighbor* (k-NN). O objetivo desse algoritmo é encontrar os k vizinhos mais próximos de um novo ponto no hiperplano e determinar sua saída (SHAI SHALEV-SHWARTZ, 2013). Na classificação, esse algoritmo determina a saída de determinado ponto através da moda, onde a classificação do vizinho que mais aparece é a saída para aquele ponto. Já na regressão a sua saída é dada pela média dos seus k -vizinhos.

A Figura 8 apresenta o exemplo de um *1-NN*, onde os pontos na imagem são denominados pontos de amostra, e a previsão de um rótulo para um novo ponto será o rótulo do ponto de amostra no centro da célula à qual ele pertence. Essas células são chamadas de Tesselação Voronoi do espaço.

Figura 8 – Exemplo de *1-NN*



FONTE: (SHAI SHALEV-SHWARTZ, 2013)

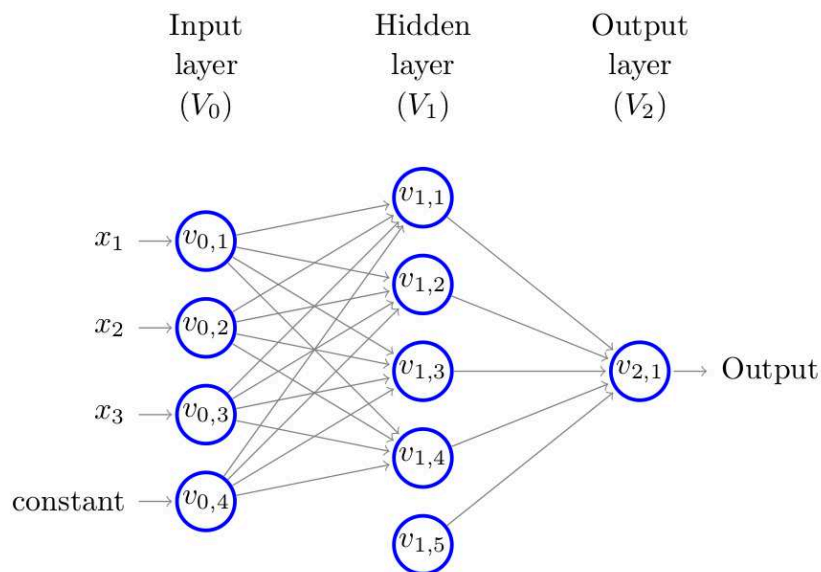
3.3.5 Artificial Neural Network

Uma rede neural artificial é um modelo inspirado pela estrutura neural cérebro. Em modelos simplificados do cérebro, consiste em um grande número de dispositivos computacionais básicos que são conectados uns aos outros em uma rede de comunicação, através da qual o cérebro é capaz de realizar cálculos altamente complexos (SHAI SHALEV-SHWARTZ, 2013). Uma Rede Neural também pode ser descrita como um grafo direcionado cujo nós correspondem a neurônios e bordas que correspondem a elos entre eles. Cada neurônio recebe como entrada uma soma ponderada das saídas dos neurônios conectados às suas bordas de entrada (SHAI SHALEV-SHWARTZ, 2013). O formato mais simples de rede neural são as *Feed forward Neural Networks*, que serão explicadas abaixo.

3.3.5.1 Feed forward Neural Networks

Uma *Feed forward Neural Network* tem por característica não conter ciclos em seu processamento, ou seja, o processamento sempre visa a camada posterior da rede. Na Figura 9 é exemplificada a estrutura básica de uma *Feed forward Neural Networks*, onde x_1, x_2, x_3 e *constant* são entradas da rede. V_0 é a camada de entrada, responsável por conter neurônios com proporção equivalente a dimensão da entrada (na figura existem quatro entradas, portanto quatro neurônios de entrada). V_1 , denominada camada oculta da rede (*hidden layer*), pode conter um número variável de neurônios (neste caso cinco). E por fim, V_2 responsável pelo resultado processado na rede. Vale ressaltar que todas as camadas que estão entre a camada de entrada e saída são denominadas camadas escondidas (SHAI SHALEV-SHWARTZ, 2013).

Figura 9 – Exemplo de rede neural



FONTE: (SHAI SHALEV-SHWARTZ, 2013)

A saída de uma *Feed forward Neural Network* é uma função de ativação aplicada à soma

ponderada da entrada mais um *bias* (adapta o conhecimento fornecido a rede neural) (DUARTE; STÅHL, 2019). A função de ativação pode ser uma função sigmoide (descrita anteriormente) ou uma outra função não linear.

Geralmente em redes neurais que possuem mais de uma camada oculta, utiliza-se a técnica de *Backpropagation*. Esse algoritmo tem por objetivo otimizar os pesos para que a rede neural possa aprender a mapear corretamente as entradas. A técnica de *Backpropagation* pode ser dividido em duas partes. A primeira, a propagação, é etapa em que as entradas são propagadas pela rede para obter as previsões de saída. A segunda etapa é o cálculo do gradiente da função de perda da rede. Utiliza-se esse gradiente para re-propagar os erros para camadas anteriores da rede até que um limiar definido como erro mínimo seja alcançado (SHAI SHALEV-SHWARTZ, 2013; ALPAYDIN, 2010).

3.3.6 *Support Vector Machine*

O algoritmo de *Support Vector Machine* (SVM) aborda o desafio de complexidade de amostra procurando por separadores. De forma simplificada, é um espaço que separa um conjunto de treinamento com uma margem, se todos os exemplos não estiverem apenas no lado correto do hiperplano de separação, mas também longe dele. Restringir o algoritmo para gerar um grande separador de margem pode render uma baixa complexidade de amostra, mesmo se a dimensionalidade do espaço de recurso for alta (e até infinita) (SHAI SHALEV-SHWARTZ, 2013).

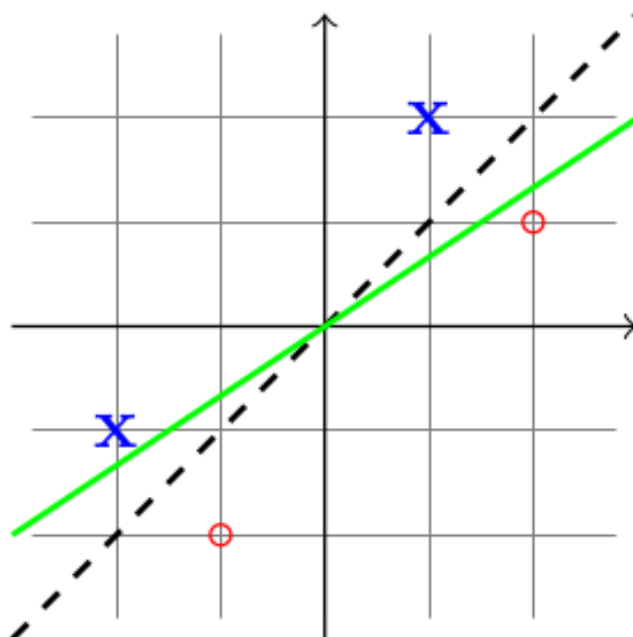
Seja $S = (x_1, y_1), \dots, (x_m, y_m)$ um conjunto de exemplos de treinamento, onde cada $x_i \in \mathbb{R}^d$ e $y_i \in \{-1, 1\}$. Pode se dizer que este conjunto de treinamento é linearmente separável, se existe um espaço, (w, b) , tal que $y_i = \text{sign}(\langle w, x_i \rangle + b)$ para todo i (SHAI SHALEV-SHWARTZ, 2013). Esta condição pode ser escrita da seguinte forma:

$$\forall i \in [m], y_i(\langle w, x_i \rangle + b) > 0$$

na Figura 10 pode-se perceber dois hiperplanos, um separado por uma reta continua e outro por uma reta pontilhada. Ambos os hiperplanos separam os quatro exemplos. Intuitivamente pode-se inferir a preferir a preferência pelo hiperplano pontilhado, pela sua maior capacidade de divisão dos exemplos. Uma maneira de formalizar essa intuição é usar o conceito de margem. A margem de um hiperplano em relação a um conjunto de treinamento é definida como a distância mínima entre um ponto no conjunto de treinamento e o hiperplano. Se um hiperplano tiver uma margem grande, ele ainda separará o conjunto de treinamento, mesmo que perturbemos levemente cada instância (SHAI SHALEV-SHWARTZ, 2013).

O SVM é um algoritmo de aprendizado no qual é retornado um hiperplano do *ERM* (minimização do erro) que separa o conjunto de treinamento com a melhor margem. Abaixo estão algumas variantes do algoritmo de SVM para classificação utilizadas nesse trabalho.

Figura 10 – Exemplo de hiperplanos de uma SVM



FONTE: (SHAI SHALEV-SHWARTZ, 2013)

- **Support Vector Classification:** mesma ideia da SVM porém sua predição é feita basicamente de forma classificatória;
- **NU Support Vector Classification:** semelhante ao SVC porém há controle sobre o número de vetores de suporte do algoritmo;
- **Linear Support Vector Classification:** também semelhante ao SVC porém o núcleo da SVM é definido como linear.

3.3.7 Gaussian Naive Bayes

O classificador *Naive Bayes* é uma demonstração clássica de como pressupostos gerativos e estimativas de parâmetros simplificam o processo de aprendizagem (SHAI SHALEV-SHWARTZ, 2013). Considere o problema de prever um rótulo $y \in \{0, 1\}$ com base em um vetor $x = (x_1, \dots, x_d)$, onde se assume que cada x_i é $\{0, 1\}$, tal que o classificador de Bayes é definido por

$$BAYES(X) = \operatorname{argmax} P[Y = y | X = x], y \in \{0, 1\}$$

Para descrever a função de probabilidade $P[Y = y | X = x]$ precisa-se de 2 parâmetros, cada um dos quais corresponde a $P[Y = 1 | X = x]$ para um certo valor de $x \in \{0, 1\}^d$ (SHAI SHALEV-SHWARTZ, 2013). Nesta abordagem presume-se de que dado o rótulo, as características são independentes uma da outra, ou seja,

$$P[X = x|Y = y] = \prod_{i=1}^d P[X_i = x_i|Y = y].$$

Simplificando,

$$BAYES(X) = \operatorname{argmax} P[Y = y] \prod_{i=1}^d P[X_i = x_i|Y = y]. \quad y \in \{0, 1\}.$$

Quando também estima-se os parâmetros usando o princípio da máxima verossimilhança, o classificador resultante é chamado de *Naive Bayes Classifier* (SHAI SHALEV-SHWARTZ, 2013). Dentro da ideia do algoritmo de *Gaussian Naive Bayes* a probabilidade das características é assumida como:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Onde os parâmetros σ_y e μ_y são estimados usando verossimilhança.

3.4 CONSIDERAÇÕES IMPORTANTES

Este capítulo introduziu conceitos e técnicas que são recorrentes na área de aprendizagem de máquina, abordando questões que foram relacionadas com o presente trabalho:

- **Dados:** como citado anteriormente, os dados podem ser divididos em conjunto de treinamento e conjunto de testes que serão objeto de predição do modelo. O conjunto de dados de treinamento foi composto por características dos campeonatos de 2016, 2017 e 2018 da Liga Nacional de Futsal do Brasil, enquanto o conjunto de teste foi composto por jogos de 2019;
- **Tipos de aprendizado:** neste capítulo foram abordados os dois principais tipos de aprendizado de máquina, o supervisionado e o não supervisionado. Neste trabalho o aprendizado foi realizado com base na aprendizagem supervisionada, pois o conjunto de treinamento possui rótulo para cada cenário, ou seja o resultado de cada jogo já é conhecido;
- **Técnicas de aprendizado:** dentro da aprendizagem supervisionada existem as técnicas de regressão e classificação, o método de classificação foi escolhida para construção dos modelos;
- **Algoritmos para criação do modelo:** o elemento central do trabalho, o modelo de predição, necessita para sua criação um algoritmo que gere um modelo matemático. Abaixo seguem técnicas utilizadas e os métodos da biblioteca *scikit-learn* da linguagem de programação *Python*, na qual os modelos foram implementados:

– *Decision Tree: DecisionTreeClassifier* (DT);

- *Random Forest: RandomForestClassifier* (RF);
- Algoritmos de *Boost: AdaBoostClassifier* (AB) e *GradientBoostingClassifier* (GB);
- *Nearest Neighbors: KNeighborsClassifier* (KN);
- *Artificial Neural Network: MLPClassifier* (MLP);
- *Support Vector Machine: SVC* (SVC), *NuSVC* (NUSVC) e *LinearSVC* (LSVC);
- *Gaussian Naive Bayes: GaussianNB* (GNB).

Como resultado do modelo de predição, será apresentado o possível resultado do jogo, seja ele vitória do time mandante, visitante ou empate. A avaliação do modelo foi dada com base nos resultados obtidos em vários cenários propostos.

4 TRABALHOS RELACIONADOS

Neste capítulo serão apresentados os trabalhos que foram destacados como relevantes, e que tenham por objetivo a previsão de resultados de partidas esportivas. Vale ressaltar que os três trabalhos apresentados abaixo são relacionados ao futebol, dado que nenhum artigo com características semelhantes direcionado ao futsal foi encontrado. Por fim, será feita a relação entre as características descritas e o presente trabalho.

4.1 USANDO CLASSIFICAÇÕES ELO PARA PREVISÃO DE RESULTADOS

O trabalho proposto em (HVATTUM; ARNTZEN, 2010) investiga o uso de classificações ELO para criar co-variáveis para modelos de previsão de resultados de partidas. O sistema de classificação ELO foi inicialmente desenvolvido para avaliar a força dos jogadores de xadrez, mas tem sido amplamente adotado em vários outros esportes, incluindo o futebol. Sistemas de classificação similares também são atualmente empregados por alguns serviços de apostas. No entanto, o uso de classificações ELO para prever resultados de partidas no futebol não tem sido extensivamente testado e avaliado usando métodos científicos.

4.1.1 Classificações ELO

O sistema ELO funciona com base nos resultados de um conjunto de correspondências anteriores. Cada equipe pode receber uma classificação de ELO como uma medida da força atual da equipe (HVATTUM; ARNTZEN, 2010).

$$y^H = \frac{1}{1 + c^{(l_0^A - l_0^H)/d}}$$

$$y^A = 1 - y^H = \frac{1}{1 + c^{(l_0^H - l_0^A)/d}}$$

O cálculo das forças das equipes pode ser dividido em três partes. A primeira é referente a y^H e y^A , que representam a média das pontuações do time mandante e visitante respectivamente para a partida. Para esse cálculo utiliza-se a classificação atual l_0^H do time mandante, l_0^A para o time visitante e as constantes c e d , que servem como escala de classificação e devem ser definidas no calibramento.

A segunda parte é definir um sistema de pontuação, onde α^H , coeficiente de pontuação do mandante receberá um valor conforme jogo da classificação atual. Segue abaixo o esquema:

$$\alpha^H = \begin{cases} 1 & \text{se time mandante venceu} \\ 0.5 & \text{se houve empate} \\ 0 & \text{se time mandante perdeu} \end{cases}$$

Com o coeficiente do time mandante calculado, cabe definir o coeficiente do time visitante, dado por $\alpha^A = 1 - \alpha^H$.

A terceira parte diz respeito ao cálculo da força dos times, que é dada da seguinte forma:

$$l_1^H = l_0^H + k(\alpha^H - y^H)$$

$$l_1^A = l_0^A + k(\alpha^A - y^A)$$

sendo l_1^H a força do time mandante, l_1^A a força do time visitante, l_0^H e l_0^A as classificações atuais de cada time, y^H e y^A as médias encontradas na primeira etapa, α^H e α^A os coeficientes de pontuação encontradas na segunda etapa e k , que é considerado a velocidade de alteração da força do time, então se k é baixo, a força do time pode demorar a se alterar, se k é alto, torna-se não confiável pelo seu nível de variação (HVATTUM; ARNTZEN, 2010).

4.1.2 Metodologia

O autor utiliza duas variantes do sistema ELO tradicional descrito acima para os testes. O primeiro, ELO_b , que trabalha de forma a utilizar como única co-variável a diferença de classificação em favor ao time mandante, dada por $x = l_0^H - l_0^A$. A segunda variante, O ELO_g , trabalha de forma muito semelhante ao ELO_b , nesse método não se utiliza o método tradicional e sim apenas a ideia de diferença para o time visitante como co-variável (HVATTUM; ARNTZEN, 2010).

Em ambos os casos, utiliza-se o modelo de *Ordered Logit Regression* como técnica principal do modelo. Esse método é semelhante a regressão logística, abordada no Capítulo 3, com extensão que essa técnica pode ser utilizada para variáveis dependentes e por consequência a co-variável gerada pelo sistema ELO.

Como forma de análise, o autor (HVATTUM; ARNTZEN, 2010) cita alguns sistemas que serão base de comparação com o ELO_b e ELO_g :

- **UNI:** este método ignora completamente qualquer informação disponível e fornece uma distribuição uniforme sobre os resultados possíveis de uma partida, dando uma probabilidade prevista de 1/3 para cada resultado (vitória em casa, empate e vitória fora);
- **FRQ:** este método leva em conta a frequência observada de cada resultado ao fazer previsões e, portanto, considera a probabilidade de uma vitória em casa igual à frequência observada de vitórias em casa em partidas anteriores;
- **GODb:** aqui as co-variáveis usadas são as 50 co-variáveis de desempenho de equipe baseadas nos resultados anteriores;
- **GODg:** neste método usa-se as 100 co-variáveis de desempenho de equipes baseadas em partidas anteriores;

- **AVG:** no AVG pega-se o resultado de diversas casas de apostas, calcula-se cada resultado das probabilidades médias e depois toma-se o inverso. Para finalmente chegar às probabilidades dos resultados. As probabilidades médias inversas são normalizadas, de modo que a soma das probabilidades sobre os três resultados seja igual a um;
- **MAX:** o MAX, é semelhante ao AVG, mas em vez de usar as probabilidades médias das casas de apostas, ele usa as probabilidades máximas para cada vitória em casa, empate e vitória fora.

4.1.3 Medidas avaliativas

Tendo em vista todos os métodos de predição descritos anteriormente, é necessário ter alguns meios de diferenciá-los em termos de qualidade. A seguir, serão descritas algumas medidas estatísticas e econômicas para a avaliação de previsões:

- **Funções de Perda:** existem várias funções de perda diferentes que podem ser usadas para avaliar métodos de previsão. Aqui concentra-se basicamente na perda quadrática e perda informacional. Em geral, a perda é uma medida obtida pela comparação entre as probabilidades de previsão e o resultado observado. A perda quadrática, também denominada L^2 mede a precisão das previsões probabilísticas, enquanto a perda informacional, denotada por L^I , corresponde a quantidade de informação necessária para comunicar resultados observados. (HVATTUM; ARNTZEN, 2010);
- **Medidas Econômicas:** considera-se três esquemas diferentes de gerenciamento financeiro. Na estratégia UNIT BET, se usa um tamanho de aposta fixo de uma unidade, que dará um ganho igual às probabilidades menos uma se a aposta for um sucesso, ou uma perda igual a uma se a aposta for uma falha. Na estratégia UNIT WIN, também leva-se as *odds* (chances) em consideração ao fazer apostas, escolhendo uma aposta de tal forma que uma aposta bem sucedida dê um ganho de uma unidade. Finalmente, assumindo que tem-se as probabilidades corretas, um tamanho ótimo de aposta pode ser definido de acordo com o critério KELLY. No que é chamado de estratégia KELLY, leva-se em conta tanto as probabilidades quanto a margem, com um tamanho de aposta igual para $(op - 1) / (o - 1)$, onde p é a probabilidade de predição e o é a probabilidade (HVATTUM; ARNTZEN, 2010);
- **Testes estatísticos:** as funções de perda e os retornos das apostas podem ser calculados para cada partida, mas uma avaliação confiável de um método de previsão requer que os cálculos sejam repetidos para um grande número de eventos, relatando o desempenho médio observado. Para cada partida, computa-se as perdas para cada um dos métodos. Um teste t de pares combinados pode então ser aplicado para testar diferenças significativas na perda média teórica para os dois métodos. O teste requer que os dois métodos sejam aplicados às mesmas correspondências (HVATTUM; ARNTZEN, 2010).

4.1.4 Principais resultados

Com os métodos propostos e técnicas de avaliação abordadas, serão relatados os resultados dos testes computacionais. Em particular, o autor se concentra no método ELO_g que parece ser o melhor dos métodos que são construídos em torno das classificações ELO.

Os testes foram feitos com base nos dados das quatro principais divisões do sistema de ligas inglesas (atualmente nomeadas *Premiership*, *Championship*, *League One* e *League Two*) em 14 temporadas, de 93/94 á 07/08, cobrindo um total de 30.524 partidas.

Na Figura 11 é apresentada uma tabela com os resultados de três estratégias de apostas diferentes (UNIT BET, UNIT WIN e KELLY). O padrão geral é que a estratégia UNIT WIN parece dar retornos (TROB) mais altos do que UNIT BET. Uma vez que o primeiro colocará apostas (BS) menores quando as probabilidades são altas. Também pode-se ver que a estratégia KELLY não se sai bem nas apostas gerais. Como essa estratégia define o tamanho das apostas assumindo que as probabilidades reais estão disponíveis, isso indica que nenhuma das previsões produzidas é competitiva quando comparada ao mercado (HVATTUM; ARNTZEN, 2010).

Figura 11 – Retornos a partir de diferentes estratégias de apostas

	#BETS	UNIT BET		UNIT WIN		KELLY	
		BS	TROB	BS	TROB	BS	TROB
UNI	27 290	1.000	0.935	0.371	0.940	0.086	0.922
FRQ	16 892	1.000	0.928	0.448	0.938	0.090	0.915
GOD _b	13 644	1.000	0.912	0.505	0.920	0.052	0.898
GOD _g	14 161	1.000	0.945	0.535	0.947	0.056	0.927
ELO _b	12 142	1.000	0.930	0.559	0.943	0.041	0.924
ELO _g	12 152	1.000	0.939	0.568	0.954	0.040	0.940
AVG	5 594	1.000	0.954	0.318	0.967	0.009	0.946

FONTE: (HVATTUM; ARNTZEN, 2010)

4.1.5 Comentários Finais

Foram implementados e testados dois métodos de previsão com base na classificação ELO para o futebol. Os métodos usam diferenças de classificação ELO como co-variáveis em modelos de regressão. Os métodos baseados em ELO foram comparados com seis métodos de previsão de *benchmark*. Foi detectado serem significativamente piores do que os dois métodos baseados em probabilidades de mercado, mas melhor do que todos os outros métodos, em termos de perda observada (HVATTUM; ARNTZEN, 2010).

Descobriu-se que as funções de perda estatística foram mais eficientes do que as medidas econômicas na diferenciação entre os vários métodos de previsão e que as classificações ELO parecem ser úteis na codificação de informações sobre resultados anteriores. No caso do futebol, a diferença de classificação única é um preditor altamente significativo de resultados de partidas (HVATTUM; ARNTZEN, 2010).

4.2 UM MODELO DE REDE BAYESIANA PARA PREVISÃO DE RESULTADOS

Uma rede bayesiana é um modelo probabilístico gráfico que representa as dependências condicionais entre variáveis incertas, que podem ser tanto objetivas quanto subjetivas (A. C. CONSTANTINOU; N. E. FENTON; NEIL, 2012).

Em (A. C. CONSTANTINOU; N. E. FENTON; NEIL, 2012) foi apresentado um novo modelo de rede bayesiana para previsão dos resultados de jogos de futebol na forma de distribuição de $p(H)$, $p(D)$, $p(A)$, correspondendo à vitória do mandante, empate e vitória do visitante, respectivamente.

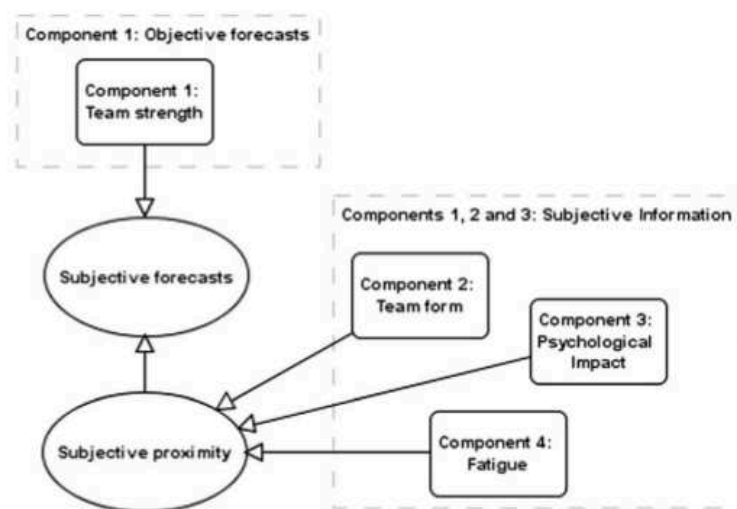
Os dados básicos utilizados para informar os antecedentes do modelo foram os resultados de todos os jogos da Premier League (EPL) da temporada 1993/94 à 2009/10 (um total de 6.244 partidas). As previsões geradas pelo modelo foram para a temporada 2010/11, em um total de 380 partidas da EPL.

O modelo, denominado *pi-football*, gera previsões para uma partida particular, considerando quatro fatores genéricos para a equipe em jogos como mandante e visitante, a força do time, forma da equipe, impactos psicológicos e desgaste (A. C. CONSTANTINOU; N. E. FENTON; NEIL, 2012):

- **Força do time:** esse fator fornece uma estimativa da força atual de cada equipe expressa como uma distribuição, usando resultados históricos das equipes, informações atuais e informações subjetivas;
- **Forma do time:** este item indica o desempenho recente da equipe específica em relação às expectativas e é medido pela comparação do desempenho esperado da equipe em relação ao desempenho observado durante as cinco semanas mais recentes;
- **Impactos psicológicos:** e psicológico de uma equipe é determinado por indicações subjetivas em relação à motivação, espírito de equipe, questões gerenciais e potencial, tal que a rede bayesiana estima a diferença no impacto psicológico entre as duas equipes;
- **Desgaste:** o fator desgaste de uma equipe é determinado pela dificuldade da partida anterior, pelo número de dias de folga, pelo número de jogadores titulares descansados e pela participação de jogadores em partidas da seleções. Assim como no item anterior a rede bayesiana estima a diferença no nível de fadiga entre as duas equipes.

A Figura 12 apresenta a relação entre os fatores citados acima em forma de componentes. Especificamente cada componente é formado por uma rede bayesiana que tem por objetivo formar uma única rede capaz de prever os resultados. Percebe-se que o componente força da equipe está diretamente ligado a previsões objetivas, enquanto os outros componentes tem uma relação subjetiva com o modelo.

Figura 12 – Conexão entre componentes



FONTE: (A. C. CONSTANTINOU; N. E. FENTON; NEIL, 2012)

Para avaliar o desempenho do método *pi-football* em (A. C. CONSTANTINOU; N. E. FENTON; NEIL, 2012) são usadas avaliações de exatidão e lucratividade, tal que ambas serão citadas a seguir:

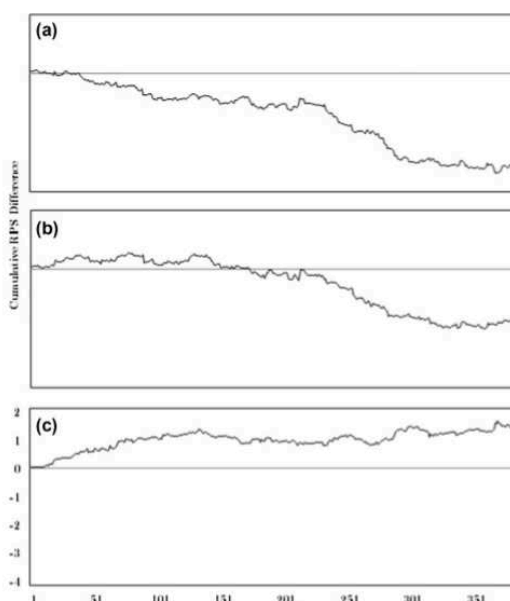
4.2.1 Medição de precisão

Para avaliar a precisão das previsões, utilizou-se o *Score* (RPS), uma regra de pontuação que tem sido descrita como particularmente apropriada na avaliação de variáveis probabilísticas de intervalo e escala ordinal. Em geral, essa regra de pontuação representa a diferença entre as distribuições acumuladas observadas e previstas, em que uma diferença maior leva a uma penalidade mais alta, que está sujeita a um viés negativo mais forte para o tamanho do pequeno conjunto (A. C. CONSTANTINOU; N. E. FENTON; NEIL, 2012).

Para determinar a precisão do modelo, calcula-se o RPS para as três tipos previsões, objetivas (fO), subjetivas (fS) e de previsões de casas de apostas (fB).

A Figura 13 apresenta a diferença cumulativa do RPS para (a) $fB - fO$, (b) $fB - fS$ e (c) $fO - fS$. Um valor de RPS mais alto indica um erro maior, ou seja, uma diferença cumulativa para $A - B$ abaixo de 0 indica que A é mais preciso que B. A partir dessa ideia, concluiu-se que as previsões objetivas foram significativamente inferiores às das casas de apostas, e que as informações subjetivas melhoram previsões de modo que estivessem no mesmo nível do desempenho das casas de apostas. Isso também sugere que as casas de apostas, como no modelo *pi-futebol*, fazem uso de informações que não são capturadas pelos dados estatísticos padrão de futebol disponíveis para o público.

Figura 13 – Diferença de RPS cumulativa



FONTE: (A. C. CONSTANTINOU; N. E. FENTON; NEIL, 2012)

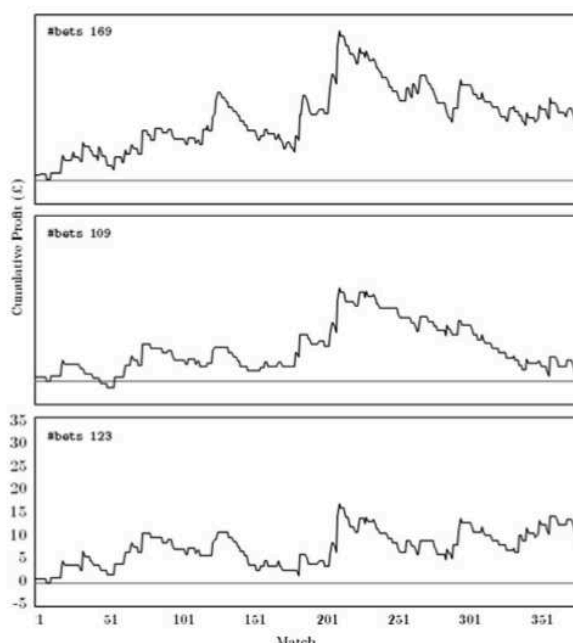
4.2.2 Medição de rentabilidade

Para determinar a lucratividade do modelo em *pi-futebol* (A. C. CONSTANTINOU; N. E. FENTON; NEIL, 2012), considera-se os seguintes três conjuntos diferentes de probabilidades de apostas:

- ***the maximum***: este conjunto de dados é usado para estimar como um apostador informado, que sabe como escolher as melhores probabilidades comparando as probabilidades das diferentes casas de apostas, poderia ter realizado sua aposta (f_m);
- ***the mean***: este conjunto de dados é usado para estimar como um apostador leigo poderia ter realizado, assumindo que ele seleciona uma casa de apostas aleatoriamente (f_a);
- ***the most common***: este conjunto de dados é usado para estimar como o apostador comum do Reino Unido poderia ter realizado. Para isso, considera-se as probabilidades proporcionadas pelo líder da casa de apostas do Reino Unido, William Hill, que representa os 25% do mercado total em todo o Reino Unido e na Irlanda (f_w).

A Figura 14 demonstra o lucro/perda cumulativo gerado contra (a) f_m , (b) f_a e (c) f_w após cada partida subsequente, assumindo uma participação de 1 libra quando a condição de aposta é satisfeita. O modelo gera um lucro em todos os três cenários e a simulação quase nunca leva a uma perda cumulativa negativa, mesmo permitindo a margem de lucro dos apostadores embutidos. No geral, o modelo *pi-football* ganhou aproximadamente 35% das apostas simuladas em todos os três cenários. Isto sugere que o modelo foi capaz de gerar lucro através de apostas de longa distância.

Figura 14 – Lucro/perda acumulado dado o fS



FONTE: (A. C. CONSTANTINOU; N. E. FENTON; NEIL, 2012)

4.2.3 Comentários Finais

Para avaliar o desempenho do modelo proposto, foi considerado medições de precisão e rentabilidade, já que estudos anteriores mostraram conclusões conflitantes entre os dois e sugerem que ambas as medições deveriam ser consideradas. Na verdade, o *pi-football* conseguiu gerar lucro em relação às probabilidades máximas, médias e comuns das casas de apostas, permitindo até mesmo a margem de lucro embutida das casas de apostas (A. C. CONSTANTINOU; N. E. FENTON; NEIL, 2012).

Mostrou-se que a informação subjetiva melhorou significativamente a capacidade de previsão do modelo. O estudo também enfatiza a importância das redes bayesianas, nas quais informações subjetivas podem ser representadas e exibidas sem qualquer esforço particular. Nenhum outro trabalho publicado parece ser particularmente bem sucedido em vencer todas as probabilidades das casas de apostas ao longo de um grande período de tempo, o que destaca o sucesso do pi-futebol (A. C. CONSTANTINOU; N. E. FENTON; NEIL, 2012).

4.3 EXPLORANDO CLASSIFICADOR POLINOMIAL PARA PREVER RESULTADOS DE PARTIDAS

Em (MARTINS et al., 2017) é introduzida uma nova abordagem para prever resultados de campeonatos de futebol. Essa abordagem investiga dados de partidas para prever os resultados, sendo elas de vitória, empate ou derrota. O sistema proposto emprega um algoritmo de classificação polinomial que analisa e define os resultados das partidas.

4.3.1 Classificador Polinomial (POL)

O classificador polinomial é um método de classificação supervisionado que apresenta resultados relevantes na análise de dados de imagens médicas, principalmente para problemas nos quais os dados não são linearmente separáveis. Este algoritmo procura expandir o espaço de recursos de entrada para uma dimensão espacial maior, de uma maneira que permita uma separação mais adequada entre as classes analisadas (MARTINS et al., 2017). Tal que a mesma está definida na equação abaixo:

$$g(x) = a^t p_n(x)$$

Onde a é a função dos coeficientes vetoriais da base polinomial, p_n é a função da base polinomial e n a ordem ou grau da função polinomial.

4.3.2 Metodologia

A abordagem proposta foi organizada em três etapas: coleta de dados, que é baseada em procedimentos relacionados aos eventos que ocorrem durante uma partida de futebol, seleção de características para classificação polinomial e classificação através do algoritmo polinomial e técnicas de aprendizado de máquina (MARTINS et al., 2017). Segue abaixo alguns pontos importantes que foram seguidos para que as etapas propostas fossem cumpridas:

- **Conjunto de dados:** os dados utilizados para a abordagem proposta foram os resultados dos jogos de futebol, obtidos nos diferentes campeonatos: *English Premier League* (EPL), temporada 2014/2015; *La Liga Primera Division* (LLPD), temporada 2014/2015; e campeonatos brasileiros, temporadas 2010 (BLC 2010) e 2012 (BLC 2012). Para os campeonatos da EPL e LLPD foram utilizadas 18 *features* para classificação, enquanto na BLC foram utilizadas 54 *features* (MARTINS et al., 2017);
- **Aprendizado de máquina:** para investigar a previsão dos resultados, alguns dos principais algoritmos de aprendizado de máquina foram utilizados. Escolhendo os módulos *NaiveBayes*, o J48, o *MultilayerPerceptron*, o módulo RBF e o módulo SMO (WEKAs da versão própria do vetor de suporte) para implementação dos classificadores *Naive Bayes* (NB), *Decision Tree* (DT), *Multilayer Perceptron* (MLP), *Radial Base Function* e *Support Vector Machine* (SVM), respectivamente (MARTINS et al., 2017);
- **Feature Selection:** para seleção de *features* foi implementado o *Principal Component Analysis* (PCA) e o *Ralief*. PCA é uma abordagem estatística que pode ser aplicada para analisar grandes conjuntos de dados multivariados, principalmente quando é frequentemente desejável reduzir a dimensionalidade. Já o *Ralief* é um algoritmo de seleção de

características baseado na ideia de estimar os recursos de acordo com o quão bem seus valores discriminam entre instâncias que estão próximas umas das outras (MARTINS et al., 2017);

- **Avaliação da abordagem:** a análise estatística foi baseada no *Student t-teste*, que é um método para medir o significado das características de discriminação entre os dados. Para avaliar a abordagem proposta, a medida de precisão AC também é considerada. A medida AC é definida como a proporção de previsões corretas relacionadas ao número total de amostras (MARTINS et al., 2017).

4.3.3 Principais resultados

A Figura 15 descreve a precisão média dos algoritmos, utilizando a técnica de *cross validation*. Vale ressaltar que a precisão é calculada com base em resultados acumulativos, ou seja, vitória/empate, empate/derrota ou vitória/derrota.

Figura 15 – Precisão média para classificação com técnica de *cross validation*

Datasets	Classifiers					
	POL	NB	DT	MLP	RBF	SVM
EPL	99.7	80.7	99.4	100.0	97.4	98.6
LLPD	99.5	83.1	99.4	100.0	97.3	93.4
BLC 2010	99.6	79.0	97.1	94.9	82.7	97.6
BLC 2012	98.9	82.0	97.0	95.2	84.3	97.2
Friedman rank	1.5	6.0	3.0	2.5	4.75	3.25
p-value	0.0103					

FONTE: (MARTINS et al., 2017)

A Figura 16 segue a mesma ideia da Figura 15, porém a técnica utilizada foi a *sliding window*. Repare em ambas as tabelas, que a precisão média do algoritmo POL, levando em consideração todas as base de dados, foi maior em comparação aos outros modelos. Outra forma de visualização dessa conta é através do *Friedman rank*, técnica probabilística que determina que quanto menor seu coeficiente melhor é sua precisão.

Figura 16 – Precisão média para classificação com técnica de *sliding window*

Datasets	Classifiers					
	POL	NB	DT	MLP	RBF	SVM
EPL	97.7	70.3	85.6	87.8	78.1	79.3
LLPD	97.5	75.1	86.1	84.5	79.3	78.6
BLC 2010	99.8	67.0	75.7	74.0	68.2	73.6
BLC 2012	99.8	66.5	76.6	71.8	64.1	71.9
Friedman rank	1.0	5.75	2.25	3.0	5.0	4.0
p-value	0.0031					

FONTE: (MARTINS et al., 2017)

4.3.4 Comentários Finais

Os dados foram amostrados utilizando as duas técnicas, *cross validation* e *sliding window*, e avaliados pelos algoritmos de classificação. A análise empírica mostra que o algoritmo POL apresentou resultados relevantes para o processo de classificação de dados. Isso resultou em valores de precisão acima de 96% para as métricas dos classificadores (MARTINS et al., 2017).

No classificador de POL, as principais características foram selecionadas durante o estágio de treinamento e teste para a definição de uma solução única, que separa as classes de forma a obter o resultado final para os grupos investigados. Estas características mais relevantes foram empregadas na análise de algoritmos clássicos de aprendizagem, permitindo uma melhoria nos níveis de precisão obtidos para os dados dos campeonatos (MARTINS et al., 2017).

O método de predição mostra-se extremamente relevante, atingindo níveis de acurácia considerados importantes para a determinação de resultados de partidas de futebol, seja diretamente com o classificador POL ou indiretamente com a seleção de características para os algoritmos de aprendizado de máquina. É importante ressaltar que os índices de acurácia obtidos foram relevantes para os dados presentes na literatura para campeonatos com um sistema de pontos acumulados (MARTINS et al., 2017).

4.4 CONSIDERAÇÕES IMPORTANTES

Dados os trabalhos citados em seções anteriores, cabe agora abordar alguns pontos importantes e que podem ser relacionados com o presente trabalho:

- **Força da equipe:** (HVATTUM; ARNTZEN, 2010) baseou-se em co-variáveis de força da equipe como base para criação do modelo de predição;
- **Algoritmos:** em (MARTINS et al., 2017) foram utilizadas técnicas de aprendizado de máquina para comparar os resultados encontrados pelo POL;
- **Features:** no contexto de seleção de características importantes para criação do modelo, cabe frisar (A. C. CONSTANTINOU; N. E. FENTON; NEIL, 2012), responsável por anexar a rede características objetivas e subjetivas para melhora da precisão. Outro trabalho importante nesse sentido, (MARTINS et al., 2017) trabalha um grande número de *features* específicas do jogo, porém o grande detalhe é a ideia de modelar o problema e as *features* conforme a competição estudada;
- **Medidas avaliativas:** tanto *cross validation*, quanto o *sliding window* foram formas importantes de avaliar o modelo em (MARTINS et al., 2017);
- **Resultados:** em relação aos resultados obtidos, pode ser destacado (MARTINS et al., 2017), que através de suas formas avaliativas conseguiu uma precisão maior em comparação com as propostas anteriores, visto os resultados superaram 96% de acerto.

As correlações citadas nesses trabalhos foram de suma importância na execução desse projeto. A ideia de força de equipe estruturou os modelos construídos, obtendo resultados satisfatórios, a relação de comparação entre algoritmos foi tomada como base para obtenção da contribuição desse trabalho. A composição de *features* com o modelo e sua criação, as medidas avaliativas do modelo dando base para utilização do *stratifiedkfold* e por fim a relação dos resultados obtidos para análise de contribuição de tais trabalhos.

5 PROJETO DOS EXPERIMENTOS

Este capítulo apresenta a metodologia utilizada para realizar os experimentos deste trabalho. A seguir serão apresentadas as etapas para a construção do conjunto de dados, as técnicas para criação do modelo e as formas de avaliação dos resultados obtidos.

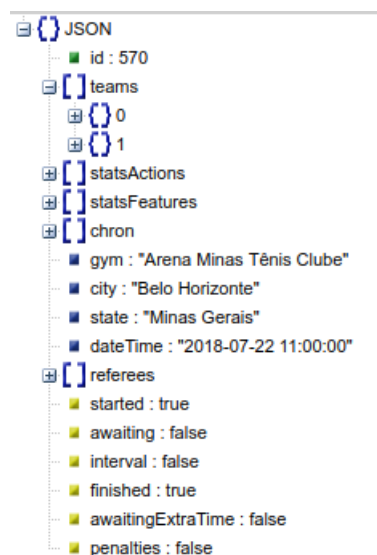
5.1 OBTENÇÃO DOS DADOS

A primeira etapa do projeto de experimentos, a obtenção dos dados de partidas da Liga Nacional de Futsal do Brasil, foi realizada através do método de extração. A execução desse método passou pela implementação de um algoritmo capaz de acessar a página web da LNF¹ e obter através desse acesso o conjunto de dados utilizado. A implementação desta etapa foi dividida nos seguintes passos:

- a) Verificação estrutural da página onde os dados estavam alocados;
- b) Carregamento das informações;
- c) Seleção de informações relevantes;
- d) Criação do conjunto de dados.

No passo (a), foram feitas análises de como o método poderia ser implementado. Verificou-se que as informações da página eram carregadas no formato *JSON* no qual esses *JSONs* estavam nomeados por números e que podiam ser extraídos em forma sequencial através de um algoritmo. A Figura 17 apresenta a estrutura do arquivo *JSON*, com possíveis campos a serem carregados na próxima etapa.

Figura 17 – Estrutura do arquivo *JSON*



¹ ligafutsal.com.br

Com acesso a estrutura da página de locação dos dados, o próximo passo (*b*) da implementação foi carregar as informações contidas no (*JSONs*) para o programa de criação do banco de dados. Esse carregamento foi executado de forma sequencial, atribuindo as informações carregadas a *dicts*, uma estrutura de dados em *Python* equivalente a *JSON*.

O passo seguinte (*c*) diz respeito a selecionar as informações possivelmente relevantes dos dados que foram carregados. Com base na ideia explorada no Capítulo 2, foram selecionadas informações referente ao código do time, número de faltas cometidas, número de cartões amarelos recebidos, número de cartões vermelhos recebidos, número de gols marcados, número de tempos técnicos pedidos, número de substituições realizadas e o ganhador da partida. Todas as informações relacionadas a estatísticas do jogo foram atribuídas tanto ao time mandante, como para o time visitante, assim como o período do jogo em que as estatísticas ocorreram.

No passo (*d*) as informações selecionadas na etapa anterior foram atribuídas a um arquivo de texto. Juntamente com essas informações foram adicionadas manualmente informações referente a rodada e fase da competição em que a partida ocorreu, pois tais características não estavam presentes no arquivo *JSON*.

O algoritmo de extração foi implementado na linguagem de programação *Python* na versão 2.7.12. O conjunto de dados representa os jogos realizados no período de 2016 a 2019, totalizando 598 partidas.

5.2 CRIAÇÃO DO MODELO

5.2.1 *Features* do Modelo

O processo de identificação das características que melhor descrevem o modelo de predição baseado no conjunto de dados obtido foi realizado através da seleção de *features* e da criação de novas *features*. Ambos os processos serão descritos abaixo.

5.2.1.1 Seleção de *Features*

Inicialmente, o conjunto de dados possuía 39 *features*, sendo 38 delas referentes aos dados do time mandante e time visitante em ambos os tempos da partida e uma *feature* que caracteriza o vencedor da partida (empate, vitória do mandante e vitória do visitante). Como o objetivo do trabalho é prever o resultado da partida baseado em estatísticas do primeiro tempo, a primeira parte da etapa de seleção foi baseada em remover as características do segundo tempo, obtendo então o número de 17 *features*, sendo dezesseis delas relacionadas a estatísticas do jogo e uma referenciando o resultado da partida (rótulo).

A etapa de seleção subsequente foi aplicar o método *SelectKBest*, da biblioteca *scikit-learn* da linguagem de programação *Python*, tal que o método funciona de maneira que, dado um número inteiro *k* e um conjunto de dados *X*, o método *SelectKBest* retorna as *k* melhores

features de X . Os valores para k nesse algoritmo variaram de um a dezesseis, visto que dezesseis é o número máximo de *features* possível, conforme etapa anterior.

Com base na classificação das melhores *features* descritas na Tabela 1, foi possível executar os algoritmos e verificar quais conjuntos de atributos tiveram melhor desempenho. Para essa verificação foram utilizados os algoritmos de classificação DT, RF, AB, KN, MLP, GB, SVC, LSVC, NUSVC, GNB, todos eles descritos no Capítulo 3. Os parâmetros utilizados nessa execução foram os *default* de cada algoritmo, com exceção ao atributo *random_state* = 5 para aqueles algoritmos que possuíam esse atributo.

Tabela 1 – Qualificação das *Features* pelo método *SelectKBest*

Qualificação	Atributo	Tipo
1	Gols no primeiro tempo time visitante	Inteiro
2	Gols no primeiro tempo time mandante	Inteiro
3	Tempos técnicos no primeiro tempo time mandante	Inteiro
4	Tempos técnicos no primeiro tempo time visitante	Inteiro
5	Código do time mandante	Inteiro
6	Substituições no primeiro tempo time visitante	Inteiro
7	Cartões amarelos no primeiro tempo time mandante	Inteiro
8	Substituições no primeiro tempo time mandante	Inteiro
9	Faltas no primeiro tempo time mandante	Inteiro
10	Cartões amarelos no primeiro tempo time visitante	Inteiro
11	Rodada da fase	Inteiro
12	Fase da competição	Inteiro
13	Cartões vermelhos no primeiro tempo time mandante	Inteiro
14	Código do time visitante	Inteiro
15	Cartões vermelhos no primeiro tempo time visitante	Inteiro
16	Faltas no primeiro tempo time visitante	Inteiro
17	Resultado	Inteiro

Outra ferramenta utilizada nessa verificação foi o método *StratifiedKfold*, também da biblioteca *scikit-learn* da linguagem de programação *Python*. Tal método tem por objetivo dividir o conjunto de dados em k *Folds*, sendo que cada *Fold* apresenta valores estratificados de cada classe, para a execução dos experimentos. Os parâmetros utilizados nesse método foram os *default*, com exceção aos parâmetros *shuffle* = *True*, responsável por embaralhar os dados de treinamento, *random_state* = 5, para fixar os valores do algoritmo e *n_splits* = 5, que significa que o conjunto de treino foi dividido em cinco partes para realização dos testes.

A Tabela 2 apresenta o valor médio de acertos em relação a execução dos algoritmos para todos os *Folds* do conjunto de treinamento, visto que a coluna *Features* representa o número das *features* utilizadas segundo o *SelectKBest*. Por exemplo, o valor 16 na coluna *Features* indica que foram utilizadas as 16 *features*, já a linha com o valor 7, indica que foram utilizadas as *features*, 1, 2, 3, 4, 5, 6 e 7 (conforme Tabela 1). Em negrito estão os algoritmos que tiveram melhor resultado para cada execução do *SelectKBest*.

Tabela 2 – Média de acertos (%) em relação a melhores atributos e algoritmos

<i>Features</i>	DT	RF	AB	KN	MLP	GB	SVC	LSVC	NUSVC	GNB
16	48,77	53,55	56,56	49,11	56,03	59,22	50,71	54,98	55,50	51,23
15	49,13	51,97	57,81	46,64	56,92	58,51	50,71	55,50	58,16	51,59
14	48,77	53,37	59,77	46,82	54,24	58,69	50,71	55,68	57,45	52,66
13	50,34	51,24	58,87	45,57	53,53	55,68	50,71	56,21	51,96	50,88
12	47,35	54,96	56,39	45,75	59,75	56,04	50,71	56,21	51,77	53,91
11	47,53	54,80	56,91	46,11	55,30	56,21	50,71	55,85	53,02	54,98
10	50,89	54,09	56,91	48,57	57,10	55,84	50,71	57,09	52,30	54,62
9	48,91	52,50	57,97	50,52	57,63	54,97	50,71	57,44	53,02	54,80
8	48,76	52,30	56,93	49,29	58,34	54,24	50,71	57,63	54,26	54,97
7	52,65	56,74	59,40	49,48	60,29	57,26	50,71	59,75	51,61	56,75
6	50,18	56,04	60,11	53,03	61,53	57,80	50,71	59,58	53,73	57,10
5	54,79	54,80	56,92	60,09	59,91	57,64	51,59	58,15	54,76	43,80
4	58,33	58,15	61,17	58,34	60,81	59,74	61,70	62,58	50,53	61,69
3	60,64	60,64	61,17	57,96	62,24	61,70	62,58	62,58	46,61	62,58
2	61,17	61,89	61,35	58,50	62,23	61,17	61,16	62,23	39,17	61,87
1	50,71	50,71	50,71	45,53	50,71	50,71	50,71	50,71	24,47	47,88

Perceba que as linhas com *Features* 2 e 3 (os dois e três melhores atributos, respectivamente) obtiveram os melhores resultados. Também pode ser percebido que comparando os resultados para os dois e três melhores atributos, o conjunto de dois melhores leva vantagem nos algoritmos DT, RF, AB e KN, enquanto os três melhores atributos possuem resultados melhores no restante dos algoritmos.

A escolha do melhor conjunto de *features* foi baseada nos resultados obtidos e apresentados na Tabela 2. Com base nisso a escolha foi pelo conjunto dos dois melhores atributos apesar do conjunto de 3 atributos ter melhor desempenho na maioria dos algoritmos. A escolha foi realizada considerando que atributo *tempo técnico* (do conjunto de 3 atributos) não teria um papel tão importante para o modelo em consideração as regras do jogo exploradas no Capítulo 2. A Tabela 3 apresenta a relação de *features* selecionadas neste passo.

Tabela 3 – Relação de *Features* selecionadas

<i>Feature</i>	Atributo	Tipo
1	Gols no primeiro tempo time visitante	Inteiro
2	Gols no primeiro tempo time mandante	Inteiro

5.2.1.2 Criação de *Features*

Após a etapa de seleção, foi realizada a etapa de criação de novos atributos com objetivo de melhorar o desempenho do modelo. Esses novos atributos foram criados com base no conjunto de dados extraído e a relação que as próprias *features* tem entre elas.

As duas primeiras *features* criadas têm como ideia identificar a força de uma equipe baseada em seu histórico. As características consideradas para a criação desses atributos foram:

- a) **Aproveitamento:** considerou-se o percentual de aproveitamento em pontos da equipe nas temporadas de 2016, 2017 e 2018;
- b) **Manutenção:** Considerou-se o percentual de jogos que a equipe estava empatando ou ganhando e conseguiu manter o resultado;
- c) **Reação:** foi considerado o percentual de jogos que a equipe estava perdendo e conseguiu empatar ou ganhar a partida e em jogos que a equipe estava empatando e conseguiu ganhar a partida;

Baseados nessas características, as *features* de força do time mandante e visitante em um jogo foi determinado por:

- Se o jogo terminar empatado na primeira etapa:
 - $forcaMandante = (aproveitamento * 3) + (manutencao * 2) + reacao;$
 - $forcaVisitante = (aproveitamento * 3) + (manutencao * 2) + reacao;$
- Se o mandante terminar ganhando a primeira etapa:
 - $forcaMandante = manutencao * 2;$
 - $forcaVisitante = reacao * 2;$
- Se o visitante terminar ganhando a primeira etapa:
 - $forcaMandante = reacao * 2;$
 - $forcaVisitante = manutencao * 2.$

A Tabela 4 mostra a média de acertos com a inclusão das *features* criadas, onde a coluna FT enumera cada uma delas. A execução foi realizada com o mesmo conjunto de dados e as mesmas configurações sobre o teste de melhores atributos. Percebe-se que os resultados com a inclusão das *features* de força do time (FT 1) foram melhores em relação ao teste com as duas melhores *features* em praticamente todos os algoritmos, com exceção a DT e RF. A média dos resultados referentes aos algoritmos com as duas melhores variáveis foi de 59.05%, enquanto o teste com a inclusão da força do time teve média de 62.94%

Outras duas *features* incluídas dizem respeito à diferença de gols ao final do primeiro tempo. A ideia por trás da criação dessas *features* é a inferir e diferenciar a dificuldade de mudança de resultado conforme a diferença de gols e força da equipe. Então a diferença de gols do time mandante e time visitante são definidas da seguinte forma:

- Se o jogo terminar empatado na primeira etapa:
 - $diferencaGolsMandante = 0;$

- $diferencaGolsVisitante = 0$;
- Se o mandante terminar ganhando a primeira etapa:
 - $diferencaGolsMandante = golsMandante - golsVisitante$;
 - $diferencaGolsVisitante = -diferencaGolsMandante$;
- Se o visitante terminar ganhando a primeira etapa:
 - $diferencaGolsVisitante = golsVisitante - golsMandante$;
 - $diferencaGolsMandante = -diferencaGolsVisitante$.

Na Tabela 4 também estão apresentados os resultados após a inclusão das *features* de diferença de gols (FT 2). Observa-se que os resultados foram melhores em relação ao teste com as duas melhores *features* em praticamente todos os algoritmos, com exceção de DT. A média dos resultados referentes aos algoritmos com as duas melhores variáveis foi de 59.05%, enquanto o teste com a inclusão da diferença de gols teve média de 62.55%, porém ficando abaixo dos resultados da inclusão da força do time, onde o resultado médio foi de 62.94%

Por fim as últimas *features* adicionadas dizem respeito à relação entre a força do time e a diferença de gols e sua interferência no poder de reação ou manutenção de um resultado. A relação diferença de gols do time mandante e visitante é definida da seguinte maneira:

- $diferencaForcaMandante = forcaMandante * diferencaGolsMandante$;
- $diferencaForcaVisitante = forcaVisitante * diferencaGolsVisitante$.

A Tabela 4 também apresenta a média de acertos com a inclusão das *features* da relação entre diferença de gols e força do time (FT 3). Percebe-se que os resultados foram melhores em relação ao teste com as duas melhores *features* em praticamente todos os algoritmos, com exceção a DT, AB e GNB. A média dos resultados referentes aos algoritmos com as duas melhores variáveis foi de 59.05%, enquanto o teste com a inclusão da relação de diferença de gols teve média de 63.70%, superando os resultados da inclusão da força do time e da diferença de gols onde o resultado médio foi de 62.94% e 62.55%, respectivamente.

Tabela 4 – Média de acertos (%) com inclusão das *features* criadas

FT	DT	RF	AB	KN	MLP	GB	SVC	LSVC	NUSVC	GNB
1	57,27	61,53	63,47	63,47	65,43	63,32	64,36	65,79	61,54	63,30
2	54,95	63,30	62,59	63,67	65,25	63,32	64,00	65,79	60,83	61,88
3	57,98	64,00	60,45	65,79	66,49	66,69	66,14	65,78	62,04	61,71

A Tabela 5 apresenta as *features* que foram utilizadas na criação do modelo, processo que será explicado posteriormente. Lembrando que os oito primeiros atributos são relacionados a estatísticas do jogo, enquanto o nono é o rótulo do modelo.

Tabela 5 – *Features* selecionadas para criação do modelo

<i>Feature</i>	Atributo	Tipo
1	Força do time mandante	Valor de ponto flutuante
2	Força do time visitante	Valor de ponto flutuante
3	Diferença de gols time mandante	Inteiro
4	Diferença de gols time visitante	Inteiro
5	Relação diferença e força time mandante	Valor de ponto flutuante
6	Relação diferença e força time visitante	Valor de ponto flutuante
7	Gols no primeiro tempo time mandante	Inteiro
8	Gols no primeiro tempo time visitante	Inteiro
9	Vencedor	Inteiro

5.2.2 Treinamento

A fase de treinamento pode ser considerada a etapa mais operacional do projeto, pois é nesta etapa em que o modelo de predição em si é criado. Genericamente, aplica-se o conjunto de dados selecionado em determinado algoritmo, de modo que isso implicará na criação de um modelo pronto para prever saídas para determinado cenário proposto. No caso específico deste projeto o conjunto de dados é criado nas etapas de extração de dados e seleção de *features*, tal que o conjunto de dados criado é aplicado nos algoritmos de DT, RF, AB, KN, MPL, GB, SVC, NUSVC, LSVC e GNB, todos eles descritos no Capítulo 3.

Um detalhe em relação aos algoritmos acima, é que eles possuem hiper-parâmetros. Esses hiper-parâmetros funcionam como uma regulação para o algoritmo, ou seja, a ideia é utilizar a melhor configuração possível para criação do modelo. A partir dessa ideia, fica o questionamento, como selecionar os melhores hiper-parâmetros? A resposta está além do conhecimento do algoritmo em que o modelo é criado. A biblioteca *scikit-learn* da linguagem de programação *Python* oferece uma ferramenta chamada *GridSearchCV*, que tem por objetivo retornar a melhor configuração possível, conforme os hiper-parâmetros selecionados e seus valores passados na função. A descrição dos parâmetros escolhidos está exibida no Apêndice A.

Com a definição do conjunto de treinamento (jogos dos campeonatos de 2016, 2017 e 2018) e os valores para os hiper-parâmetros dos algoritmos, o próximo passo é a configuração do *GridSearchCV*. A configuração utilizada foi a *default* da função, com exceção ao parâmetro CV, que indica o número de partes que o conjunto de treinamento irá ser dividido para executar as verificações de configuração. O apêndice B apresenta os melhores hiper-parâmetros calculados para cada algoritmo.

5.3 AVALIAÇÃO DO MODELO

Após a criação do modelo e da execução dos experimentos, coube avaliar os resultados obtidos, e para isso utilizou-se algumas métricas padrões. A Tabela 6 apresenta uma versão

binária de uma matriz de confusão. Essa matriz apresenta informações sobre o modelo de predição, tal que a partir dela algumas informações podem ser extraídas. Repare que os valores de VP e VN, representam o acerto na classe determinada positiva e negativa, enquanto os valores FP e FN representam o erro de predição das classes. Sempre lembrando que essa matriz pode ser estendida para um problema multi classe, na qual este trabalho se qualifica.

Tabela 6 – Matriz de confusão

	Positivo	Negativo
Positivo	Verdadeiro Positivo (VP)	Falso Positivo (FP)
Negativo	Falso Negativo (FN)	Verdadeiro Negativo (VN)

Baseado na ideia da matriz de confusão, existem algumas métricas capazes de avaliar o modelo. Uma delas, a acurácia tem como objetivo basicamente inferir o percentual de acerto da predição. Deste modo, a acurácia pode ser representada da seguinte forma:

$$accuracy = \frac{VP + VN}{VP + VN + FP + FN}$$

Outra métrica avaliativa, a precisão, classifica determinada classe com base nos valores verdadeiros previstos para determinada classe em relação a todos os valores preditos para a mesma classe. Quanto mais próximo resultado for de 1, maior a capacidade do modelo prever corretamente valores para esta classe. Segue fórmula baseada na matriz de confusão que caracteriza essa métrica:

$$precision = \frac{VP}{VP + FP}$$

Com fórmula semelhante a precisão, a revocação também classifica a classe com base em relação aos valores verdadeiros previstos, porém sua relação é dada em função dos valores que são dessa classe e foram classificadas de forma errada. Quanto mais próximo o resultado for de 1, maior a capacidade do modelo prever valores para essa classe. A fórmula abaixo apresenta o cálculo dessa métrica:

$$recall = \frac{VP}{VP + FN}$$

Já a f_1 -score faz uma harmonia entre a precisão e a revocação e funciona como um parecer geral sobre o modelo baseado nessas duas métricas. Em especial, quanto maior o resultado, melhor o modelo tem capacidade de prever diferentes classes. Segue fórmula:

$$f_1 - score = \frac{2 * precision * recall}{precision + recall}$$

Este trabalho utilizou as métricas apresentadas nessa seção como forma avaliativa dos resultados. No Capítulo 6 a aplicação desses métodos será explicada assim como os resultados obtidos.

6 EXPERIMENTOS E RESULTADOS

Após a etapa de criação do projeto de experimentos, o próximo passo foi a realização dos mesmos. Neste capítulo serão apresentados os componentes necessários para execução dos experimentos, além da exibição dos resultados obtidos e a análise feita com base nos critérios adotados no Capítulo 5.

Como citado anteriormente, o conjunto de dados utilizado nesse trabalho é composto por 598 partidas da Liga Nacional de Futsal do Brasil, sendo elas realizadas no período de 2016 a 2019. Para realização dos experimentos, tal conjunto foi dividido em duas partes, conjunto de dados de treinamento e conjunto de dados de teste.

O conjunto de dados de treinamento foi composto por partidas dos campeonatos de 2016, 2017 e 2018, totalizando assim 564 partidas. Tal conjunto foi responsável pela criação dos dez modelos propostos para este trabalho. Já o conjunto de dados de teste foi formado por partidas realizadas no ano de 2019 e compreende os dados relativos à execução dos algoritmos e obtenção dos resultados. A Tabela 7 exibe os times participantes do campeonato, e por consequência, os integrantes das partidas em que os experimentos foram realizados.

Tabela 7 – Relação de times do campeonato 2019

Sigla	Nome
ACBF	Carlos Barbosa
ATL	Atlântico
MAG	Magnus
TUB	Tubarão
COR	Corinthians
MOU	Campo Mourão
PAT	Pato Futsal
CAS	Cascavel
FOZ	Foz Cataratas
JEC	Joinville
ASS	Assoeva
JOA	Joaçaba
COP	Copagril
MTC	Minas
MAR	Marreco
INT	Intelli
JAR	Jaraguá
SJF	São José
BLU	Blumenau

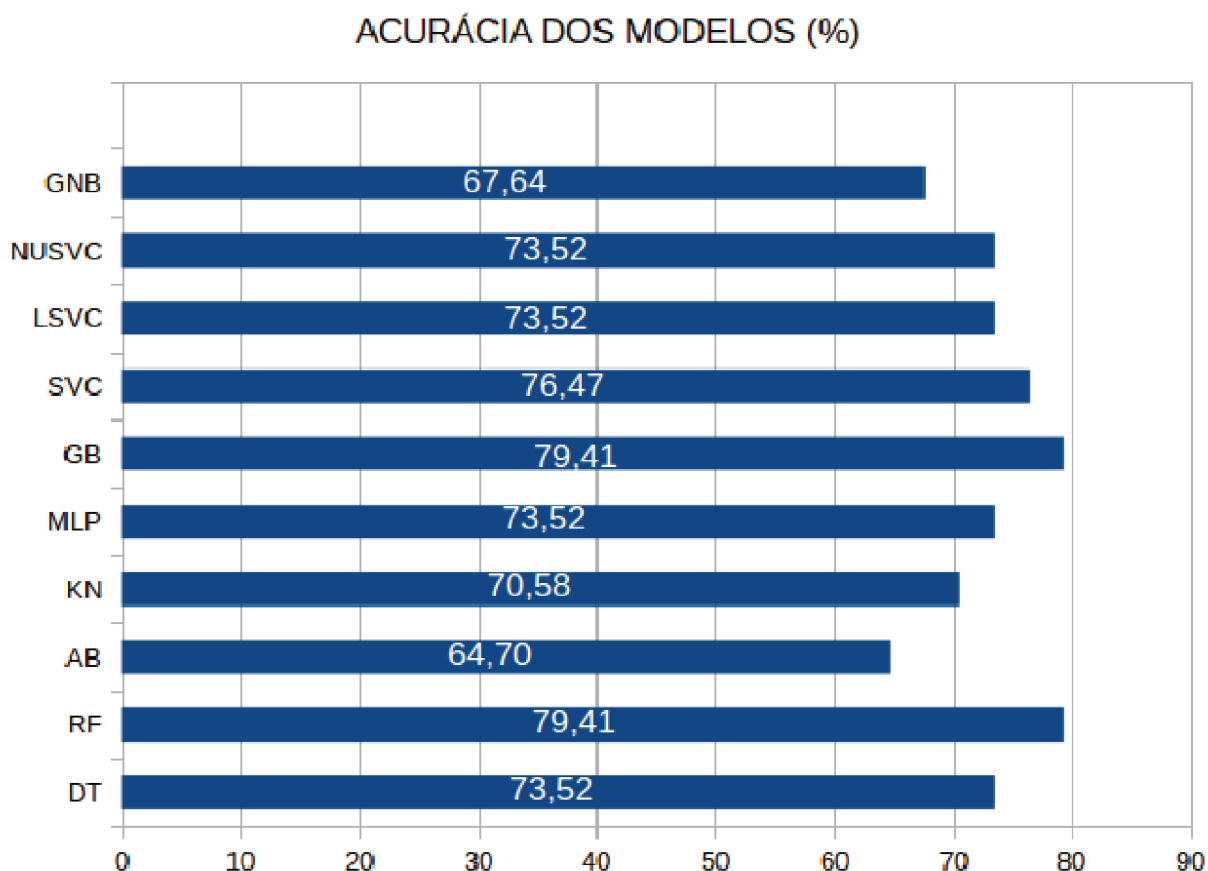
A Tabela 8 apresenta os trinta e quatro jogos do campeonato que formaram o conjunto de teste dos modelos desenvolvidos. Perceba que a tabela possui além do time mandante e visitante, um campo referenciando o resultado da partida, sendo que essas estatísticas serão usadas para avaliação dos resultados.

Tabela 8 – Relação de jogos preditos

Jogo	Time Mandante	Time Visitante	Resultado
1	ATL	INT	Mandante
2	COP	ASS	Mandante
3	BLU	MTC	Visitante
4	ACBF	MOU	Mandante
5	JAR	TUB	Visitante
6	COR	CAS	Mandante
7	PAT	SJF	Mandante
8	MAG	FOZ	Mandante
9	JOA	MAR	Empate
10	INT	ACBF	Empate
11	TUB	COR	Mandante
12	SJF	ATL	Visitante
13	MAR	COP	Mandante
14	FOZ	JOA	Mandante
15	ASS	JAR	Mandante
16	MOU	JEC	Mandante
17	CAS	PAT	Visitante
18	MTC	MAG	Visitante
19	JAR	MAR	Mandante
20	COP	FOZ	Visitante
21	ACBF	SJF	Mandante
22	BLU	MOU	Visitante
23	JEC	INT	Empate
24	ATL	CAS	Visitante
25	PAT	TUB	Visitante
26	JOA	MTC	Mandante
27	ASS	PAT	Mandante
28	FOZ	JAR	Mandante
29	INT	BLU	Mandante
30	TUB	ATL	Mandante
31	MTC	COP	Mandante
32	CAS	ACBF	Visitante
33	MAR	COR	Visitante
34	MOU	MAG	Mandante

O gráfico da Figura 18 mostra a acurácia geral dos experimentos realizados com base nos dez modelos propostos para este trabalho. Percebe-se que os algoritmos de *Gradient Boosting* e *Random Forest* obtiveram os melhores resultados (79,41%), enquanto o algoritmo de *Ada Boost* obteve o pior desempenho com uma acurácia de 64,7%. Observa-se também que todos os algoritmos tiveram desempenho bem acima da probabilidade randômica de acerto dos resultados, ou seja, 33,33% e que a média geral de todos os algoritmos foi de 73,22%, mais próximo ao resultado da *Gradient Boosting* e *Random Forest* do que do pior resultado individual, o *Ada Boost*.

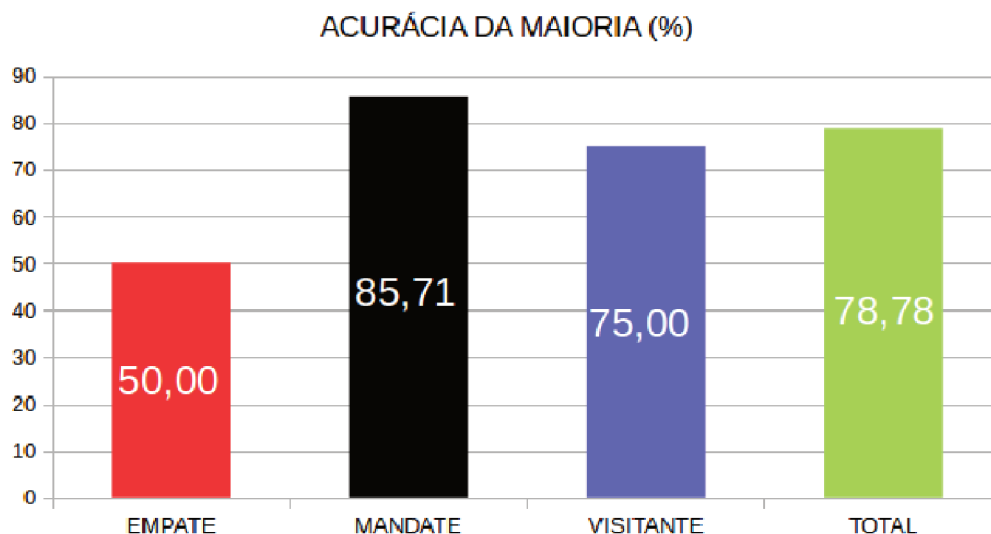
Figura 18 – Acurácia dos modelos



Além da acurácia geral dos modelos, outros tipos de análises foram feitas com intuito de perceber situações em que os modelos utilizados em conjunto podem ter melhor desempenho para prever resultados de partidas. Abaixo serão apresentadas algumas análises propostas com base nos experimentos realizados, tendo em vista o uso de comitês de modelos e associações que os mesmos podem ter.

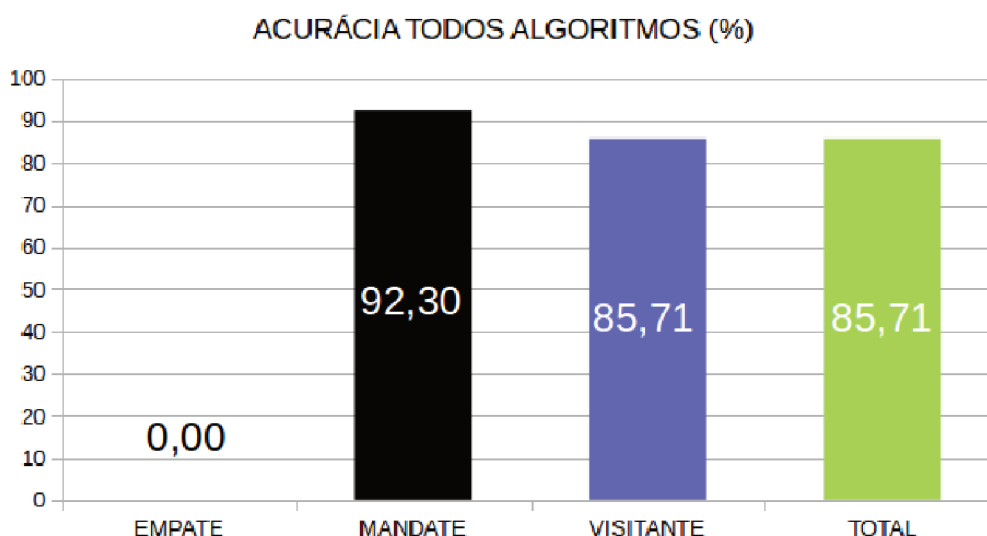
Na ideia de comitês, outra forma avaliativa está relacionado a acurácia dos modelos em situações em que a maioria previu o mesmo resultado. O gráfico da Figura 19 exibe esta situação. Para esta análise, utilizou-se todos os dados de experimento com exceção ao jogo 28, citado na Tabela 8, na qual não houve maioria absoluta. Dos trinta e três jogos utilizados, vinte e um tiveram como maioria de predição o time mandante como vencedor, oito o time visitante como vencedor e quatro como empate. Nota-se que os melhores resultados foram em casos que a maioria previu o time mandante como vencedor, com a acurácia de 85,71%, em contrapartida o pior desempenho ficou no empate com a acurácia de 50%, sempre levando em consideração que existem apenas quatro jogos preditos como empate, ou seja, o modelo acertou metade. A média de acertos nesse tipo de avaliação levando em conta a maioria absoluta de predições teve como resultado 78,78%, muito próximo aos resultados do melhor índice individual (79,41%) e bem acima da probabilidade randômica de 33,33%.

Figura 19 – Acurácia da maioria



Muito semelhante à análise feita em relação à maioria absoluta, o gráfico da Figura 20 teve como forma de análise situações em que todos os algoritmos tiveram a mesma predição, com base nos dados de teste, o conjunto se restringiu a vinte e uma partidas. Dessas partidas, uma teve todos os algoritmos apontando empate, treze prevendo vitória do mandante e sete prevendo vitória do visitante. Perceba que os melhores resultados novamente estiveram na classe mandante (92,30%), o pior resultado esteve novamente na classe empate, na qual, errou o único item previsto. A média de acertos nesse tipo de avaliação, tendo como base a mesma predição para todos os algoritmos, teve como acurácia 85,71%, acima do maior índice individual e muito superior a probabilidade randômica.

Figura 20 – Acurácia todos



Como exibido no Capítulo 5, existem outras métricas avaliativas para os modelos de predição. A Tabela 9 mostra os resultados referente à precisão dos dez modelos utilizados no

experimento e os dois comitês propostos (MAIORIA e TODOS). A coluna Média representa a média ponderada de precisão de cada algoritmo. Por exemplo, em DT a Média é calculada da seguinte forma: $(0,33 * NE + 0,85 * NM + 0,64 * NV)/NJ$, onde NE, NM, NV e NJ são números de empate, vitória mandante, vitória visitante e jogos, respectivamente. Verifica-se que a classe empate possui um nível bem abaixo em relação as outras duas classes. Esse baixo índice pode ser explicado com o poder de variância que a classe tem, pois a fronteira de divisão de classe com as outras varia exclusivamente com um gol, enquanto as outras classes tendem a variar bem menos, dependendo da diferença de gols que existe entre os times. Individualmente destaca-se a classe mandante do algoritmo *Gaussian Naive Bayes* que obteve resultado de 0,94. Na média geral, a melhor precisão esteve no comitê TODOS, Com 0,86, no qual esse comitê tem como análise apenas jogos em que todos os modelos tiveram a mesma predição.

Tabela 9 – Métrica *precision* (%)

Classe	Empate	Mandante	Visitante	Média
DT	0,33	0,81	0,70	0,73
RF	0,50	0,83	0,86	0,81
AB	0,20	0,82	0,86	0,78
KN	0,20	0,85	0,67	0,73
MLP	0,20	0,86	0,75	0,76
GB	0,50	0,83	0,86	0,81
SVC	0,50	0,82	0,70	0,75
LSVC	0,33	0,81	0,70	0,73
NUSVC	0,33	0,81	0,86	0,78
GNB	0,12	0,94	0,70	0,79
MAIORIA	0,50	0,81	0,75	0,76
TODOS	0,00	0,92	0,86	0,86

Outra métrica apresentada no Capítulo 5 tem papel importante na avaliação do modelo, a revocação. A revocação dos experimentos é apresentada na Tabela 10, na qual a coluna Média representa a média ponderada de revocação de cada algoritmo, calculada de forma idêntica a Tabela 9. Assim como na precisão, a classe empate possui uma revocação bem inferior à classe mandante (devido a seu grau de variância), porém verifica-se também que a classe visitante possui algumas previsões de vitória de mandante em outras classes, o que a deixa em proximidade a classe empate. O destaque nessa métrica fica para a classe mandante, onde os resultados variaram entre 0,7 e 0,95, bem acima da revocação média que esteve entre 0,65 e 0,86. Individualmente, destaca-se na classe mandante os Algoritmos de *Random Forest* e *Gradient Boosting*, com valor de 0,95. Com melhor revocação média, o comitê TODOS, novamente obteve os melhores resultados.

A métrica f_1 -score faz relação entre as métricas de precisão e revocação para dar um parecer geral sobre o modelo. A Tabela 11 apresenta os resultados encontrados após os experimentos, na qual a coluna Média representa a média ponderada do *score* de cada algoritmo, calculada de forma idêntica as métricas anteriores. A classe empate é a que tem o

Tabela 10 – Métrica *recall* (%)

Classe	Empate	Mandante	Visitante	Média
DT	0,33	0,85	0,64	0,74
RF	0,67	0,95	0,55	0,79
AB	0,67	0,70	0,55	0,65
KN	0,33	0,85	0,55	0,71
MLP	0,33	0,90	0,55	0,74
GB	0,67	0,95	0,55	0,79
SVC	0,33	0,90	0,64	0,76
LSVC	0,33	0,85	0,64	0,74
NUSVC	0,67	0,85	0,55	0,74
GNB	0,33	0,75	0,64	0,68
MAIORIA	0,67	0,89	0,55	0,76
TODOS	0,00	0,92	0,86	0,86

menor potencial de prever corretamente um resultado, enquanto a classe mandante se destaca pelos bons resultados em toda a análise. No aspecto individual, destaca-se negativamente o algoritmo de *Gaussian Naive Bayes* com um *score* de 0,18 para classe empate. Na média os modelos tiveram um bom *score* variando de 0,69 a 0,86. Destaque individual para os algoritmos de *Random Forest* e *Gradient Boosting* com *score* de 0,79 e o comitê TODOS, com *score* 0,86.

Tabela 11 – Métrica *f1-score* (%)

Classe	Empate	Mandante	Visitante	Média
DT	0,33	0,83	0,67	0,73
RF	0,57	0,88	0,67	0,79
AB	0,31	0,76	0,67	0,69
KN	0,25	0,85	0,60	0,72
MLP	0,25	0,88	0,63	0,74
GB	0,57	0,88	0,67	0,79
SVC	0,40	0,86	0,67	0,76
LSVC	0,33	0,67	0,67	0,73
NUSVC	0,44	0,83	0,67	0,74
GNB	0,18	0,83	0,83	0,72
MAIORIA	0,57	0,85	0,63	0,75
TODOS	0,00	0,92	0,86	0,86

Se a análise for realizada de forma individual, percebe-se um bom desempenho do modelo em relação a sua acurácia, porém os estudos feitos com base em conjunto de modelos e comitês trouxeram aspectos a serem analisados, por exemplo, foi observado que as predições no caso em que todos os modelos previstos tiveram a mesma predição, obtiveram resultados melhores que os resultados individuais, e de forma mais específica, predições em que todos os algoritmos deram como resultado a vitória do mandante tiveram resultados ainda melhores, o que indica que o modelo tem maior facilidade em prever situações favoráveis ao time mandante.

7 CONCLUSÃO

Este trabalho teve com objetivo principal a predição de resultados de partidas da Liga Nacional de Futsal do Brasil. Durante a criação e a execução dos experimentos foram realizadas algumas conclusões importantes sobre a contribuição deste projeto.

Constatou-se que aspectos históricos das equipes influenciam diretamente em resultados de partidas de futsal e que as *features* criadas com base nesses aspectos históricos melhoraram o desempenho dos modelos. Além disso, pode-se dizer que estatísticas do jogo podem ser secundárias durante a criação desse modelo.

Em relação à acurácia geral dos modelos, o resultado foi satisfatório. Todos os dez algoritmos tiveram desempenho muito superior à probabilidade geral de 1/3, destacando-se os algoritmos de *Random Forest* e *Gradient Boosting* que alcançaram 79,41%. As análises feitas em relação aos casos da utilização de comitês com ideia de classificação por maioria absoluta e quando todos os algoritmos apresentam a mesma predição tiveram bons resultados, especialmente para a classe mandante.

As métricas avaliativas expressaram claramente como cada classe se comportou durante a etapa de experimentos. A classe empate, a mais crítica, foi identificada como classe com menor poder de predição e que isso pode ser dado pelo fato da variância que essa classe pode ter durante o jogo. A classe visitante teve desempenho superior a classe empate, porém percebe-se a dificuldade de predição em alguns casos. Esse fato pode ter algumas explicações, porém podem ser apresentadas com mais exatidão em trabalhos futuros. E por fim a classe mandante, a detentora dos melhores resultados em todos os métodos avaliativos, intuitivamente mostrando o poder de predição nessa classe.

Analisando os algoritmos individualmente, podem ser realizadas algumas observações. O algoritmo *Ada boost* teve o pior desempenho da predição geral e o pior *score* em relação as métricas avaliativas. Positivamente, desta-se o algoritmo *Gradient Boosting* e *Randon Forest* pelo melhor desempenho individual e *score* das métricas de avaliação.

Como projetos futuros relacionados a esse trabalho, pode-se verificar a ideia da criação de um modelo em tempo real para predição de resultados. Esse trabalho se dará com base na ideia da criação de *features* que equilibrem o poder de reação e manutenção do resultado dentro do jogo, levando em consideração o peso dos gols na previsão. Outro projeto pode estar relacionado a associação de modelos, tal que a utilização de uma nova técnica de aprendizado de máquina pode facilitar na análise do poder de predição de classes do modelo.

REFERÊNCIAS

- ALPAYDIN, Ethem. **Introduction to Machine Learning**. 2. ed. Massachusetts: The MIT Press, 2010. 537 p. ISBN 9780262012430.
- ANGELINI, Giovanni; DE ANGELIS, Luca. PARX model for football match predictions. **Journal of Forecasting**, Wiley Online Library, v. 36, n. 7, p. 795–807, 2017.
- BUNKER, Rory P; THABTAH, Fadi. A machine learning framework for sport result prediction. **Applied Computing and Informatics**, Elsevier, 2017.
- CONSTANTINOU, Anthony C. Dolores: A model that predicts football match outcomes from all over the world. **Machine Learning**, Springer, p. 1–27, 2018.
- CONSTANTINOU, Anthony C; FENTON, Norman E; NEIL, Martin. pi-football: A Bayesian network model for forecasting Association Football match outcomes. **Knowledge-Based Systems**, Elsevier, v. 36, p. 322–339, 2012.
- CONSTANTINOU, Anthony; FENTON, NORMAN. Towards Smart-Data: Improving predictive accuracy in long-term football team performance. **Knowledge-Based Systems**, Elsevier, v. 124, p. 93–104, 2017.
- DUARTE, Denio; STÅHL, Niclas. Machine learning: a concise overview. In: DATA Science in Practice. [S.l.]: Springer, 2019. p. 27–58.
- FRIEDMAN, Jerome H. Stochastic gradient boosting. **Computational statistics & data analysis**, Elsevier, v. 38, n. 4, p. 367–378, 2002.
- FUTEBOL DE SALÃO - CBFS, Confederação Brasileira de. **Futsal Origem O esporte da bola pesada que virou uma paixão**. 2018. Disponível em: <<http://www.cbfs.com.br/2015/futsal/origem/index.html>>. Acesso em: 25 nov. 2018.
- FUTEBOL DE SALÃO CBFS, Confederação Brasileira de. **Futsal Livro Nacional de Regras 2018**. 2018. Disponível em: <http://www.cbfs.com.br/2015/futsal/regras/livro_nacional_de_regras_2018.pdf>. Acesso em: 25 nov. 2018.
- FUTSAL, Liga Nacional de. **Classificação**. 2017. Disponível em: <<https://ligafutsal.com.br/classificacao/>>. Acesso em: 25 nov. 2018.
- HASTIE, Trevor et al. Multi-class adaboost. **Statistics and its Interface**, International Press of Boston, v. 2, n. 3, p. 349–360, 2009.
- HUANG, Kou-Yuan; CHANG, Wen-Lung. A neural network method for prediction of 2006 world cup football game. In: IEEE. NEURAL Networks (IJCNN), The 2010 International Joint Conference on. [S.l.: s.n.], 2010. p. 1–8.
- HVATTUM, Lars Magnus; ARNTZEN, Halvard. Using ELO ratings for match result prediction in association football. **International Journal of forecasting**, Elsevier, v. 26, n. 3, p. 460–470, 2010.

JOSEPH, Anito; FENTON, Norman E; NEIL, Martin. Predicting football results using Bayesian nets and other machine learning techniques. **Knowledge-Based Systems**, Elsevier, v. 19, n. 7, p. 544–553, 2006.

MARTINS, Rodrigo G et al. Exploring polynomial classifier to predict match results in football championships. **Expert Systems with Applications**, Elsevier, v. 83, p. 79–93, 2017.

SHAI SHALEV-SHWARTZ, Shai Ben-David. **Understanding Machine Learning**. 1. ed. Cambridge University Press: Cambridge University Press, 2013. 449 p. ISBN 9781107057135.

APÊNDICE A – PARÂMETROS

Segue abaixo hiper-parâmetros e seus possíveis valores selecionados em cada um dos algoritmos descritos:

- **Decision Tree:**

- *criterion*: este hiper-parâmetro seleciona uma função para medir a qualidade de uma divisão. Parâmetros escolhidos: *gini* e *entropy*;
- *splitter*: este hiper-parâmetro seleciona uma estratégia a ser usada para escolher a divisão em cada nó. Parâmetros escolhidos: *best* e *random*;
- *max_depth*: este hiper-parâmetro define a profundidade máxima da árvore. Parâmetros escolhidos: *None*, 2, 3, 4, 5, 6, 7 e 8;
- *min_samples_split*: este hiper-parâmetro define o número mínimo de amostras necessárias para dividir um nó interno. Parâmetros escolhidos: 2, 3, 4, 5 e 6;
- *min_samples_leaf*: este hiper-parâmetro define o número mínimo de amostras necessárias para estar em um nó folha. Parâmetros escolhidos: 1, 2, 3, 4 e 5;
- *max_leaf_nodes*: este hiper-parâmetro define o número máximo de nós folha. Parâmetros escolhidos: *None*, 2, 3 e 4;
- *min_impurity_decrease*: Este hiper-parâmetro seleciona o mínimo valor de impureza de um nó. Parâmetros escolhidos: 0, 1, 2 e 3;
- *presort*: Se este hiper-parâmetro for *True* ordena os dados para acelerar a descoberta de melhores divisões no ajuste. Parâmetros escolhidos: *True* e *False*.

- **Random Forest:**

- *n_estimators*: este hiper-parâmetro define o número de árvores na floresta. Parâmetros escolhidos: 5, 10 e 15;
- *criterion*: este hiper-parâmetro seleciona uma função para medir a qualidade de uma divisão. Parâmetros escolhidos: *gini* e *entropy*;
- *max_depth*: este hiper-parâmetro define a profundidade máxima da árvore. Parâmetros escolhidos: [None,2,3,4,5];
- *min_samples_split*: este hiper-parâmetro define o número mínimo de amostras necessárias para dividir um nó interno. Parâmetros escolhidos: 2, 3 e 4;
- *min_samples_leaf*: este hiper-parâmetro define o número mínimo de amostras necessárias para estar em um nó da folha. Parâmetros escolhidos: [1, 2, 3 e 4;
- *max_leaf_nodes*: este hiper-parâmetro define o número máximo de nós folha. Parâmetros escolhidos: *None*, 2, 3 e 4;

- *bootstrap*: este hiper-parâmetro se definido como *True* as amostras de *bootstrap* são usadas ao construir árvores. Parâmetros escolhidos: [*True* e *False*];
- *min_impurit_decrease*: Este hiper-parâmetro seleciona o mínimo valor de impureza de um nó. Parâmetros escolhidos: 0, 1, 2 e 3;
- *warm_start*: este hiper-parâmetro quando configurado para *True*, reutiliza a solução da chamada anterior para ajustar e adicionar mais árvores ao conjunto. Parâmetros escolhidos: *True* e *False*.

- **Ada Boost:**

- *n_estimators*: este hiper-parâmetro define o número máximo de estimadores do algoritmo. Parâmetros escolhidos: 20, 25, 50, 75 e 100;
- *learning_rate*: este hiper-parâmetro define a taxa de aprendizado do algoritmo. Parâmetros escolhidos: 0.5, 0.75, 1, 1.5 e 2;
- *algorithm*: este hiper-parâmetro seleciona o algoritmo de impulsão. Parâmetros escolhidos: *SAMME* e *SAMME.R*.

- **K-Neighbors:**

- *n_neighbors*: este hiper-parâmetro define o número de vizinhos utilizados. Parâmetros escolhidos: 2, 3, 4, 5, 6, 7 e 8;
- *algorithm*: este hiper-parâmetro define o algoritmo usado para calcular os vizinhos mais próximos. Parâmetros escolhidos: *auto*, *ball_tree*, *kd_tree* e *brute*;
- *leaf_size*: este hiper-parâmetro define o tamanho da folha passado para o algoritmo de vizinhos mais próximos. Parâmetros escolhidos: 5, 10, 20, 30, 40, 50 e 60;
- *p*: este hiper-parâmetro define a potência para a métrica *Minkowski*. Parâmetros escolhidos: 1, 2, 3, 4, 5 e 6;
- *weights*: este hiper-parâmetro seleciona a função de peso usada na previsão. Parâmetros escolhidos: *uniform* e *distance*.

- **Multi Layer Perceptron:**

- *hidden_layer_sizes*: este hiper-parâmetro define o numero de neurônios da camada escondida. Parâmetros escolhidos: 50, 75, 100, 125 e 150;
- *activation*: este hiper-parâmetro seleciona a função de ativação para a camada oculta. Parâmetros escolhidos: *identity*, *logistic*, *tanh* e *relu*;
- *alpha*: este hiper-parâmetro define o parâmetro de penalidade. Parâmetros escolhidos: 0.0001, 0.0003 e 0.0005;
- *solver*: este hiper-parâmetro define a função de otimização de pesos. Parâmetros escolhidos: *lbfgs*, *sgd* e *adam*;

- *learning_rate*: este hiper-parâmetro define a taxa de aprendizagem. Parâmetros escolhidos: *constant*, *invscaling* e *adaptive*;
- *learning_rate_init*: este hiper-parâmetro define a taxa de aprendizagem inicial. Parâmetros escolhidos: 0.001, 0.003 e 0.005;
- *max_iter*: este hiper-parâmetro define o número máximo de iterações para convergência. Parâmetros escolhidos: 100, 200 e 300;
- *momentum*: este hiper-parâmetro define a atualização do gradiente descendente. Parâmetros escolhidos: 0.5, 0.7 e 0.9;
- *beta_1*: este hiper-parâmetro define a taxa de decaimento exponencial para estimativas da primeira etapa de otimização. Parâmetros escolhidos: 0.5, 0.7 e 0.9;
- *beta_2*: a taxa de decaimento exponencial para estimativas para segunda etapa de otimização. Parâmetros escolhidos: 0.5, 0.7 e 0.999.

- **Gradient Boosting:**

- *learning_rate*: este hiper-parâmetro define a taxa de aprendizagem. Parâmetros escolhidos: 0.05, 0.1 e 0.5;
- *n_estimators*: este hiper-parâmetro define o número de estágios de reforço a serem executados. Parâmetros escolhidos: 50, 100 e 150;
- *subsample*: este hiper-parâmetro define a fração de amostras a ser usada para ajustar os exemplos a serem aprendidos. Parâmetros escolhidos: 0.1, 0.5 e 1.0;
- *min_samples_split*: este hiper-parâmetro define o número mínimo de amostras necessárias para dividir um nó interno. Parâmetros escolhidos: 2, 3 e 4;
- *min_samples_leaf*: este hiper-parâmetro define o número mínimo de amostras necessárias para estar em um nó da folha. Parâmetros escolhidos: 1, 2 e 3;
- *max_depth*: este hiper-parâmetro define a profundidade máxima dos estimadores. Parâmetros escolhidos: 2, 3 e 4;
- *min_impurity_decrease*: este hiper-parâmetro define o mínimo valor de impureza de um nó. Parâmetros escolhidos: 0, 1 e 2;
- *max_leaf_nodes*: este hiper-parâmetro define o número máximo de nó folha. Parâmetros escolhidos: *None*, 2 e 3.

- **Support Vector Classification:**

- *C*: este hiper-parâmetro define a penalidade C do termo de erro. Parâmetros escolhidos: 0.5, 1.0 e 1.5;
- *kernel*: este hiper-parâmetro especifica o tipo de núcleo a ser usado no algoritmo. Parâmetros escolhidos: *linear*, *poly*, *rbf* e *sigmoid*;

- *shrinking*: este hiper-parâmetro se *True* define o uso da heurística de encolhimento. Parâmetros escolhidos: *True* e *False*;
- *probability*: este hiper-parâmetro se *True* habilita estimativas de probabilidade. Parâmetros escolhidos: *True* e *False*;
- *decision_function_shape*: este hiper-parâmetro define a função de decisão. Parâmetros escolhidos: *ovo* e *ovr*.

- ***Linear Support Vector Classification:***

- *dual*: este hiper-parâmetro se *True* define a utilização do algoritmo de otimização. Parâmetros escolhidos: *True* e *False*;
- *C*: este hiper-parâmetro define a penalidade C do termo de erro. Parâmetros escolhidos: 0.5, 1.0 e 1.5;
- *multi_class*: este hiper-parâmetro determina a estratégia de várias classes. Parâmetros escolhidos: *ovr* e *crammer_singer*;
- *fit_intercept*: este hiper-parâmetro se *True* calcula a interceptação para este modelo. Parâmetros escolhidos: *True* e *False*;
- *intercept_scaling*: este hiper-parâmetro define o dimensionamento da interceptação para este modelo. Parâmetros escolhidos: 0.5, 1 e 1.5].

- ***Nu Support Vector Classification:***

- *kernel*: este hiper-parâmetro especifica o tipo de núcleo a ser usado no algoritmo. Parâmetros escolhidos: *poly*, *rbf*, *sigmoid*;
- *degree*: este hiper-parâmetro define o grau da função do núcleo. Parâmetros escolhidos: 2, 3 e 4;
- *shrinking*: este hiper-parâmetro se *True* define o uso da heurística de encolhimento. Parâmetros escolhidos: *True* e *False*;
- *probability*: este hiper-parâmetro *True* habilita estimativas de probabilidade. Parâmetros escolhidos: *True* e *False*;
- *decision_function_shape*: este hiper-parâmetro define a função de decisão. Parâmetros escolhidos: [*ovo* e *ovr*].

- ***Gaussian Naive Bayes:*** este algoritmo não teve hiper-parâmetros selecionados.

APÊNDICE B – PARÂMETROS SELECIONADOS

Segue abaixo melhores parâmetros selecionados após a execução do *GridSearchCV* no conjunto de treinamento:

- ***Decision Tree:***

- *min_samples_split* = 4;
- *splitter* = *random*;
- *presort* = *True*;
- *max_depth* = 3;
- *max_leaf_nodes* = *None*;
- *min_impurity_decrease* = 0;
- *min_samples_leaf* = 1;
- *criterion* = *entropy*;

- ***Random Forest:***

- *min_samples_split* = 3;
- *max_leaf_nodes* = *None*;
- *n_estimators* = 15;
- *warm_start* = *True*;
- *min_impurity_decrease* = 0;
- *min_samples_leaf* = 4;
- *criterion* = *gini*;
- *max_depth* = 5;
- *bootstrap* = *False*;

- ***Ada Boost:***

- *algorithm* = *SAMME*;
- *n_estimators* = 25;
- *learning_rate* = 0.75;

- ***K-Neighbors:***

- *leaf_size* = 5;
- *algorithm* = *auto*;

- $p = 4$;
- $weights = distance$;
- $n_neighbors = 7$;

- **Multi Layer Perceptron:**

- $learning_rate_init = 0.001$;
- $beta_2 = 0.999$;
- $solver = lbfgs$;
- $beta_1 = 0.5$;
- $activation = tanh$;
- $max_iter = 300$;
- $hidden_layer_sizes = 50$;
- $momentum = 0.5$;
- $alpha = 0.0001$;
- $learning_rate = invscaling$;

- **Gradient Boosting:**

- $min_samples_split = 3$;
- $min_samples_leaf = 2$;
- $max_depth = 4$;
- $max_leaf_nodes = 2$;
- $n_estimators = 100$;
- $min_impurity_decrease = 1$;
- $subsample = 0.5$; $learning_rate = 0.05$;

- **Support Vector Classification:**

- $shrinking = True$;
- $decision_function_shape = ovo$;
- $probability = True$;
- $kernel = sigmoid$;
- $C = 1.0$;

- **Linear Support Vector Classification:**

- $fit_intercept = True$;

- *dual = True;*
- *multi_class = ovr;*
- *C = 0.5;*
- *intercept_scaling = 0.5;*

- ***Nu Support Vector Classification:***

- *kernel = rbf;*
- *degree = 2;*
- *decision_function_shape = ovo;*
- *shrinking = True;*
- *probability = True.*