PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

Modelos de previsão para jogos de futebol

Felipe Gustavo Pereira Viberti

Projeto Final de Graduação

Centro Tecnico Cientifico - CTC

Departamento de Informatica - DI

Curso de Graduação em Ciência da Computação

Rio de Janeiro, Dezembro de 2019

Felipe Gustavo Pereira Viberti

Modelos de Previsão Para Jogos de Futebol

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

Orientador : Marco Serpa Molinaro

Rio de Janeiro, Dezembro de 2019

Resumo

Viberti, Felipe. Molinaro, Marco.**Modelos de previsão para jogos de futebol.** Rio de Janeiro, 2019, 28p. Relatório de Projeto Final 2. Pontifícia Universidade Católica do Rio de Janeiro.

Este projeto final analisa diversos modelos de aprendizado de máquina aplicados ao caso da previsão de jogos de futebol. Além dos modelos tradicionais, estamos propondo aqui um modelo híbrido que combina alguns dos outros. Analisaremos e iremos comparar os resultados de acurácia dos modelos para tentar determinar qual deles funciona melhor para o problema proposto.

Palavras Chave

Aprendizado de Máquina, Modelos de Previsão, Redes Neurais Recorrentes, LSTM

Abstract

Viberti, Felipe. Molinaro, Marco.**Prediction Models for Soccer Matches.** Rio de Janeiro, 2019, 28p. Relatório de Projeto Final 2. Pontifícia Universidade Católica do Rio de Janeiro.

This final project analyzes some machine learning models applied to the problem of predicting the outcome of soccer matches. Besides from the more traditional models we are proposing here a model that combines some aspects from two existing models. We will analyze and compare the accuracy results from the models to try to determine which one of them works better with this problem.

Keywords

Machine Learning, Prediction Models, Recurrent Neural Networks, LSTM

Lista de Figuras

Lista de Tabelas

Sumário

1.Introdução

Esportes mexem com a paixão de muitas pessoas, e por isso muitas vezes é difícil se ter uma visão lógica sobre os jogos. Porém, percebe-se um crescimento na tentativa de se fazer previsões sobre os resultados utilizando técnicas quantitativas.

Esse é um mercado que movimenta muito dinheiro pois está diretamente ligado ao mercado de apostas. Diversas técnicas quantitativas foram utilizadas para fazer previsões em esportes. Uma dessas técnicas é utilizar uma métrica de gols esperados em um jogo (xG). Essa técnica foi proposta pela primeira vez por MacDonald [1] para avaliar a performance de equipes da liga de hockey no gelo americana (NHL). Ela combina a quantidade de chutes a gol, para fora e chutes bloqueados para calcular quantos gols uma equipe deveria fazer em uma partida. Isso permite avaliar o quão efetiva é uma equipe, a partir da quantidade de gols que ela converte dado as oportunidades que ela teve. Alguns outros autores como Lucey [2] e Eggles [3] levaram essa métrica para o futebol com alterações. Esses dois autores buscaram prever a probabilidade de cada chute ser convertido em gol. Eggles, em especial utilizou modelos de aprendizado de máquina como árvores de decisão, regressão logística e Random Forest para classificar cada chance de gol em diferentes classes de probabilidade do gol acontecer. Alguns dos algoritmos explorados por Eggles serão utilizados neste trabalho mas com aplicações diferentes. Um dos exemplos mais recentes da aplicação da técnica xG foi do site FiveThirtyEight [4] que combinou essa técnica com um sistema próprio de ranqueamento ofensivo e defensivo das equipes. Esse ranking era baseado em quão boa ou ruim uma equipe era atacando e defendendo. A união dessas técnicas produziu resultados bastante interessantes [4].

Outra técnica que foi usada neste problema é a técnica de ELO Rating. Essa pontuação foi inventada por Elo [5] para avaliar jogadores de xadrez e foi levada para outros esportes por outros autores. Em 2010 Hvattuma utilizou um sistema baseado em ELO [6] para tentar prever jogos de futebol de duas formas. Primeiro, levando em conta o resultado de um jogo (vitória, empate, derrota) e depois tentando prever o placar exato da partida. Em 2013 Fenton [7] inventou um novo rating chamado de pi-rating que dá pontuações dinâmicas às equipes baseado na diferença do placar de seus jogos ao longo do tempo. Em 2018 Herbinet [8] uniu as técnicas de ELO rating e xG para gerar um modelo de classificação para prever o resultado da partida e um modelo de regressão para prever o placar da mesma.

Vale destacar que, além das técnicas, existe uma grande quantidade de material disponível para o público geral. Entre as ferramentas disponíveis está o site Football Data que apresenta estatísticas completas sobre as partidas que ocorreram desde a década de 90 nos principais campeonatos do mundo [9]. Esse site é atualizado uma vez por ano, ao final de cada temporada, com as informações de todas as partidas da temporada que terminou. Existem também APIs que possuem dados do campeonato atual e informações de quais os próximos jogos que irão ocorrer. Um exemplo é a API Football [10] que apresenta dados de 425 campeonatos pelo mundo. Outro fator que pode ser relevante para uma partida é quais jogadores estão suspensos ou machucados e não poderão participar do jogo. O site Injuries and Suspensions [11] apresenta essas informações. Saber a qualidade dos jogadores de uma equipe pode ser outro fator fundamental para a previsão e isso pode ser obtido na forma de uma nota no site Who Scored [12]. Essa nota avalia todas as contribuições que um jogador pode dar como gols, assistências, desarmes, interceptações e defesas no caso de um goleiro. É possível que a união de algumas dessas informações como a nota de um jogador e o fato de ele não poder jogar aquele jogo afetem a previsão do mesmo.

1.1 Objetivos do Trabalho

Os objetivos desse trabalho foram estudar técnicas gerais de aprendizado de máquina, e propor modelos para previsão de jogos de futebol. Apresentaremos mais detalhes sobre ambas essas direções.

1.1.1 Estudos Conceituais

Ao longo do primeiro semestre do ano foram estudados os fundamentos de diversas técnicas de aprendizado de máquina. Esses estudos foram baseados na realização de um curso online sobre o tema [13] e em apresentações periódicas para o orientador. Isso inclui os estudos dos modelos regressão linear, regressão logística, K-Nearest Neighbors, SVM, Kernel SVM, Naive Bayes, árvore de decisão, random forest e k-Means Clustering.

Para auxiliar no foco deste relatório, não descreveremos em mais detalhes esse estudo.

1.1.2 Proposta de modelo para previsão de resultados de futebol

Propomos modelos baseados em redes neurais para a previsão de resultados de futebol. O modelo híbrido final utiliza tanto uma parte não-sequencial que utiliza redes neurais tradicionais quanto uma parte sequencial que utiliza redes neurais recorrentes. Além disso, realizamos experimentos

computacionais comparando esses modelos propostos com outros modelos tradicionais. Em geral, os modelos tradicionais não sequencias obtiveram desempenho superior ao modelo híbrido mas ele foi bastante superior à um modelo que utiliza apenas redes neurais recorrentes para realizar as previsões.

2. Aprendizado de Máquina

Este capitulo apresenta alguns dos conceitos estudados e que serão aplicados no projeto.

O objetivo geral do aprendizado de máquina é treinar um modelo com dados para que ele consiga reconhecer padrões existentes e, tipicamente, fazer previsões para um conjunto de dados não vistos que tenham propriedades similares aos dados de treino. Nesse trabalho focaremos em métodos de aprendizado de máquina supervisionados, que são aqueles onde cada dado de entrada pode ser mapeado para um valor ou classe desejada. Nesse caso cada dado é um par da forma (x,y) , onde x é um vetor de atributos (features) e y é o valor ou classe desse dado. Em alto nível, em um problema de aprendizado de máquina supervisionado utilizam-se dados onde ambos x e y são conhecidos para treinar o modelo. Na presença de um novo dado, onde apenas seus atributos x são conhecidos, o modelo é utilizado para realizar uma previsão $\hat{y}$ do seu valor real y. A qualidade do modelo é tipicamente medida pela sua acurácia, definida como a razão do número de exemplos que o modelo previu corretamente dividido pela quantidade total de exemplos

2.1 Training, Validation e Test Set

Overfitting ocorre quando o modelo se ajusta demais aos dados disponíveis e perde sua capacidade de generalizar para dados não vistos. Para garantir essa capacidade de generalização, é importante separar os dados em 3 grupos: *training, validation e test set.* O training set é o que será usado no modelo para de fato treiná-lo. O validation set é usado para estimar a acurácia do modelo produzido e guiar a escolha de seus hiperparâmetros (por exemplo, o tamanho e topologia de uma rede neural, veja Seção 2.2.5). Caso os resultados da validação não sejam bons deve-se voltar ao modelo, fazer as alterações necessárias, treiná-lo novamente e voltar à validá-lo. O test set só deve ser usado quanto o modelo tiver passado pela fase da validação. Ele é o que vai determinar a acurácia do seu modelo. Depois que o modelo for aplicado no test set, não é mais possível voltar para realizar correções no mesmo. A presença de uma grande diferença na acurácia do modelo entre o training set e o validation set indica que está ocorrendo overfitting.

2.2 Modelos

Nesta seção abordaremos brevemente modelos de aprendizado de máquina utilizados nesse trabalho. Outros modelos foram estudados durante a realização do trabalho (k-Nearest Neighbors, SVM, Naive Bayes, regressão logística), mas decidimos omiti-los para melhor direcionar a leitura.

2.2.1 Árvores de Decisão

A árvore de decisão é uma representação de uma tabela de decisão na forma de árvore. Rokach [14] entra em maiores detalhes sobre árvores de decisão e pode ser consultado para aspectos específicos deste modelo. Para os propósitos deste trabalho ilustraremos através de um exemplo o funcionamento de árvores de decisão. Suponha que queremos classificar algumas observações nas classes verde e vermelha. Para fazer essa classificação temos duas variáveis numéricas X1 e X2. A classe que cada uma das observações pertence depende do valor dessas variáveis. Temos então uma possível árvore de decisão para este problema.

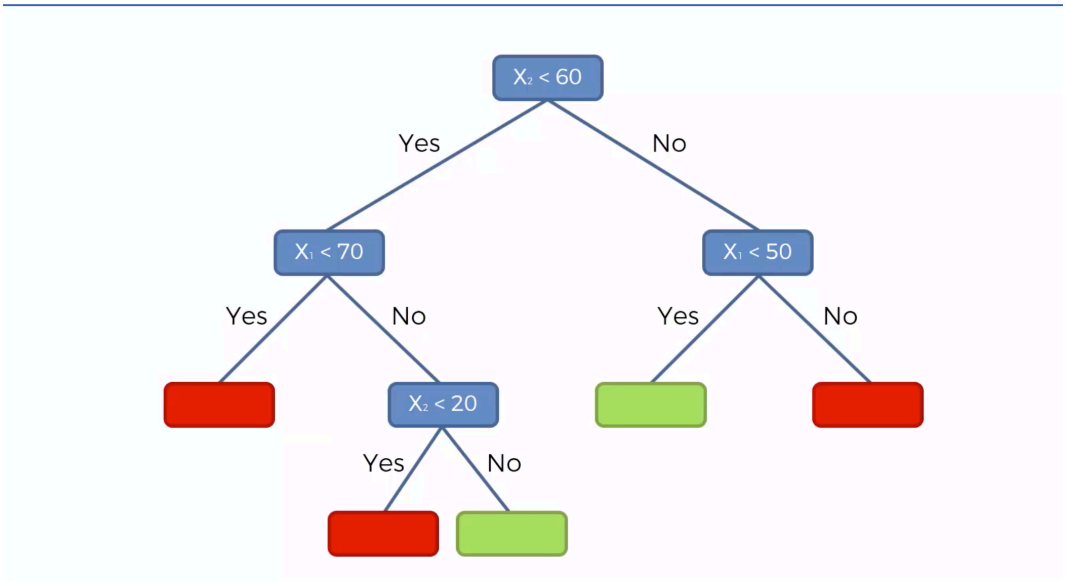


Figura 1 - Árvore de Decisão para classificação [13]

Com base nos valores dos atributos X1 e X2 obtemos a classificação seguindo a árvore de decisão na ramificação apropriada. A Tabela 1 dá alguns exemplos de valores de X1 e X2 e a classificação obtida pela árvore.

Tabela 1 - Resultados do exemplo de árvore de decisão

X1	X2	Classe
30	50	Vermelha
80	50	Verde
80	10	Vermelha
30	70	Verde
80	70	Vermelha

Para se construir uma árvore de decisão, deve-se decidir quais atributos farão parte das tomadas de decisão e em qual ordem isso será feito. Essas decisões são feitas de forma top-down, ou seja, decidindo primeiro qual atributo será utilizado na raiz da árvore e construindo o restante da árvore de forma recursiva. Intuitivamente, devemos escolher o primeiro atributo tal que os exemplos de cada uma de suas ramificações sejam os mais homogêneos possível , idealmente sendo todos da mesma classe (no qual caso esse único nó raiz é suficiente para a classificação correta dos dados). Vale destacar o importante algoritmo ID3 que utiliza entropia como medida de heterogeneidade dos exemplos em cada ramo. Peng [15] entra em detalhes sobre a implementação do algoritmo ID3.

2.2.1.1 Random Forest

O que esse modelo faz é construir uma quantidade x de árvores de decisão e dar como resultado a classe prevista pela maioria das árvores [16]. O random no nome do algoritmo vem do fato da escolha dos pontos iniciais para a construção das x árvores de decisão serem randômicos. Cada subconjunto de pontos escolhidos aleatoriamente é utilizado para criar cada uma das árvores. Essa escolha dos pontos é feita através de uma variação da técnica de bagging chamada de feature bagging [16]. Esse modelo costuma ter excelentes resultados em vários domínios; em particular, o jogo Kinect da Microsoft usa esse modelo para fazer o reconhecimento dos movimentos dos usuários [17].

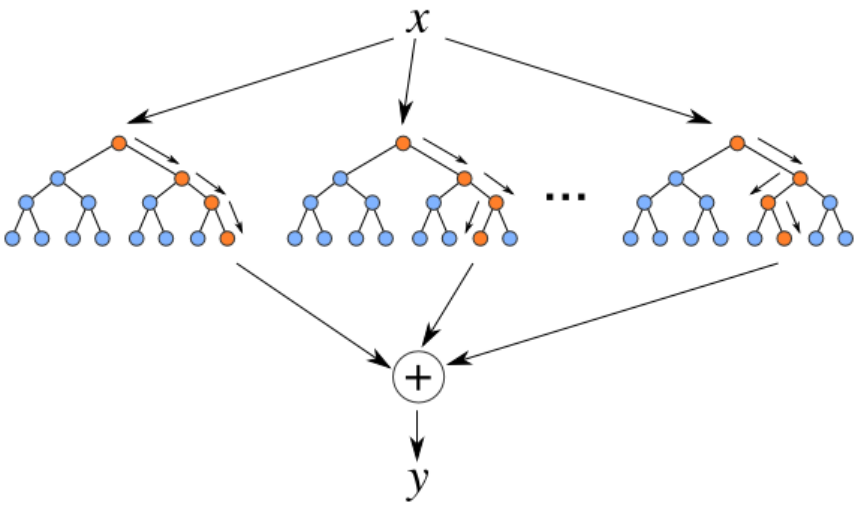


Figura 2 - Exemplo de funcionamento do random forest

O exemplo da Figura 2 é executado em cada uma das árvores e o resultado de cada uma entra numa votação majoritária para decidir a qual classe o dado pertence.

2.2.2 Rede Neural

Redes neurais são um dos modelos atualmente mais utilizados e são bastante úteis em diversas aplicações [18]. O principio básico é a criação de camadas de neurônios que passem informações para as próximas camadas. Os neurônios são conectados através de pesos que modulam a informação que é passada de um para o outro. Mais precisamente um neurônio recebe o valor de saída de outros neurônios, multiplica cada valor pelo peso associado a essa conexão, soma os valores resultantes, e executa uma função de ativação sobre essa soma, produzindo finalmente sua saída. Uma das funções de ativação mais populares é a função rectifier. Ela segue a forma abaixo:

$$f(x) = \{ 0 \text{ se } x < 0; x \text{ se } x >= 0 \}$$

Uma das principais vantagens dessa função é que ela não ativa todos os neurônios da rede ao mesmo tempo. Pela fórmula acima, se um valor for negativo a função atribui o valor zero à esse neurônio significando que ele não está ativado. Isso torna a rede neural efetivamente mais esparsa e mais eficiente para computação.

O treinamento de uma rede neural consiste na construção dos pesos entre os neurônios. Esses pesos são tipicamente inicializador de forma aleatória. Os atributos de um dado são passados como entrada da rede atual e são propagados até produzir uma previsão na saída da rede. Essa saída é comparada como o valor real que desejávamos prever, e mede-se o erro dessa

previsão por uma função de custo. A próxima etapa é o back propagation que consiste em atualizar os pesos de cada neurônio da rede baseado no erro obtido, tipicamente utilizando métodos baseados no gradiente descendente [19]. Depois que todo o training set tiver passado pela rede, temos uma epoch de treino completa. Podem ser feitas quantas epochs se queira mas geralmente o processo termina quando uma nova epoch não aumentou suficientemente a acurácia do modelo. Exemplificaremos o uso de uma rede neural em um problema concreto para facilitar a compreensão. Suponha que queremos prever o preço de um apartamento. Para isso temos dados da área do apartamento, quantidade de quartos, localização e quantidade de anos que o prédio possui. A rede neural vai tentar encontrar padrões entre essas informações. Na figura 3 vemos que o primeiro neurônio da segunda camada combina as informações de área do apartamento com distância da cidade e cada um dos subsequentes neurônios combina alguma dessas quatro informações. A previsão final é resultado de tudo que foi aprendido pela rede a partir dessas combinações intermediárias dos neurônios.

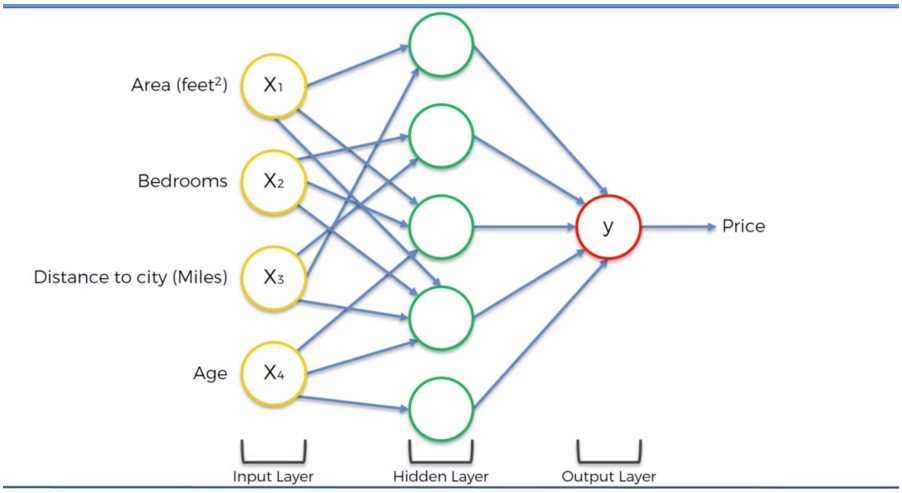


Figura 3 - Exemplo de rede neural para previsão de preços de apartamentos [13]

2.2.3 Rede Neural Recorrente

As redes neurais recorrentes (RNN) são similares às redes neurais descritas anteriormente mas com modificações que as melhor adequam a lidar com dados sequencias , em especial com séries temporais. Isso acontece porque elas tem uma memória interna que permite considerar não somente as entradas dos tempos atuais mas também entradas dos tempos anteriores para tomar as decisões.

Para tornar isso mais concerto, existem três topologias principais de RNN. A figura 4 exemplifica estes tipos. Vale destacar que a representação one-to-one que aparece nessa figura é apenas uma representação do que seria um processamento normal sem usar RNN, não sendo, portanto um tipo de RNN.

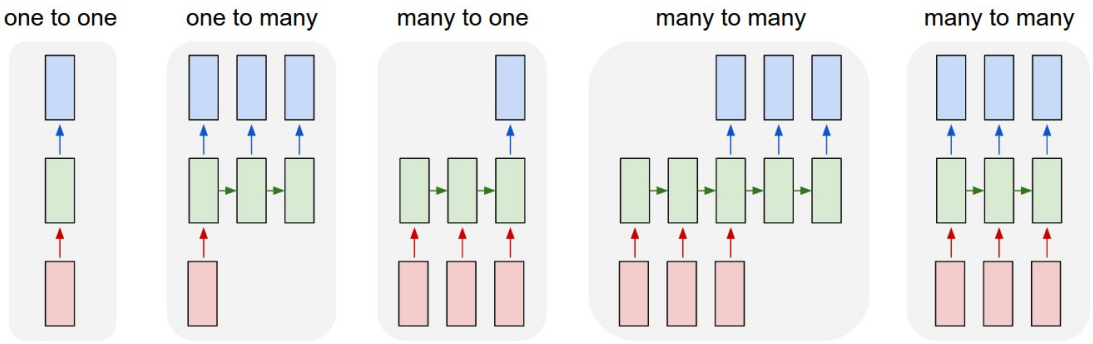


Figura 4 - Tipos de RNN [20]

One-to-Many

Nas redes one-to-many, existe apenas uma entrada que ocorre no inicio do processamento mas para cada período de tempo existe uma saída. Um exemplo desse uso é um algoritmo que receba uma imagem e tenha que gerar uma frase para legendar essa imagem. Esse algoritmo somente terá uma entrada mas em cada momento de tempo ele vai produzir uma palavra para essa frase.

Many-to-One

Nas redes many-to-one só existe uma saída no final da rede, apesar dos inputs entrarem em cada momento do tempo. Nesse caso, a entrada para a rede é uma sequência mas só se quer gerar um resultado único. Um exemplo prático disso são algoritmos de análise de sentimento onde uma frase (sequência de palavras) é passada como entrada e a saída é um sentimento único.

Many-to-Many

Nas redes many-to-many para cada momento de tempo em que existe uma entrada existe uma saída. Existem dois tipos de redes many-to-many. Na primeira vale integralmente esse principio de que para em todos os momentos de tempo tem que existir uma saída. Essas redes são chamadas de many-to-many full sync (imagem mais à direita da figura 4). Podem ser usadas para reconhecimento e tradução de voz para texto, onde cada palavra falada tem que de fato ser escrita. No segundo tipo, as chamadas semi-sync, uma sequência de

entradas é passada e no tempo em que a ultima entrada for passada, a saída começa a ser produzida. Poderiam ser usadas, por exemplo para fazer tradução de frases onde cada momento de tempo pode ser visto como uma nova palavra na frase que necessita de uma tradução. Por se tratar de uma tradução não seria possível faze-la imediatamente como no caso das redes full sync já que uma tradução de várias palavras pode ser feita apenas por uma por exemplo. Além disso, as traduções dependem muito do contexto e de entradas anteriores.

2.2.3.1 LSTM

Long Term Short Term Memory (LSTM) é um tipo especial de rede neural recorrente capaz de aprender dependências por um longo período de tempo. Todas as redes neurais recorrentes tem a forma de uma cadeia de módulos de repetição, onde cada módulo de repetição é uma rede neural. Para conseguir apresentar esse comportamento os módulos de repetição nas LSTM têm quatro camadas internas que interagem entre si.

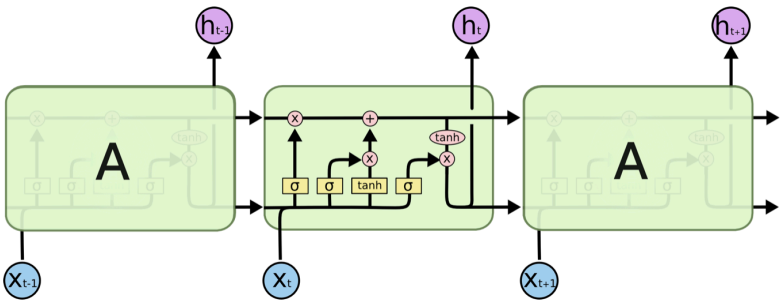


Figura 5 - Estrutura interna LSTM [21]

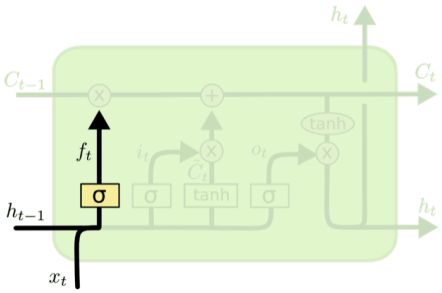
Ideia Geral

A ideia chave dessa estrutura (Figura 5) passa muito pela linha superior que corre reta por toda a célula. Ela é chamada de cell state e corre livre pela célula, podendo ser alterada somente por estruturas alto nível do LSTM chamadas de gates. Todos os gates são compostos por uma camada de rede neural que usa a função sigmoide e por uma operação de multiplicação nos vetores. A função sigmoide sempre produz valores entre zero e um que neste contexto significam o quanto aquele gate deve deixar passar de informação, onde o valor zero significa que ele não deve deixar nada passar e o valor um significa que ele deve deixar tudo passar.

Passo a Passo

Forget Gate

A primeira etapa é decidir que informações serão descartadas do cell state. Isso é feito através de uma função sigmoide que olha para o output da camada anterior (ht-1) e para o input da cada atual (xt) e gera um número entre 0 e 1 para cada número existente no cell state (Ct-1). Baseado nesse número esse gate sabe o que ele deve deixar passar e o que deve ser descartado.

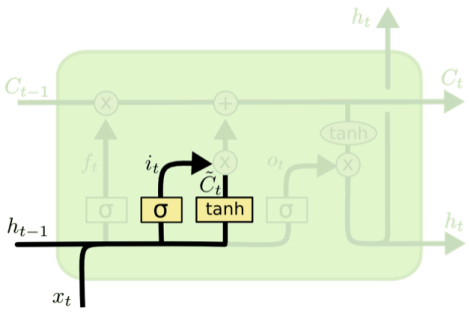


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Figura 6 - Representação e Fórmula Forget Gate [21]

Input Gate

A próxima etapa nesse processo é decidir que novas informações iremos armazenar no cell state. Esse processo tem duas partes. Primeiro, o input gate decide que valores serão atualizados e depois uma camada com a função hiperbólica tangente (tanh) cria um vetor de valores candidatos que podem ser adicionados no estado. Esses dois vetores serão combinados na próxima etapa para produzir uma atualização para o cell state.



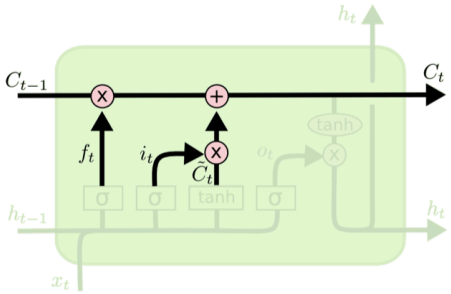
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Figura 7 - Representação e Fórmula Input Gate [21]

Atualização do state cell

As etapas anteriores revelaram o que tem que ser feito e nessa etapa é onde de fato isso é executado. Primeiro, multiplicamos o valor antigo do estado (Ct-1) pelo resultado do forget gate para descartar os valores necessários.

Depois adicionamos a isso, a multiplicação do resultado do input gate com o vetor de valores candidatos, também produzido na etapa anterior ($it * Ct$). Esse é o novo vetor de valores candidatos, agora já escalado por quanto decidimos atualizar cada valor.

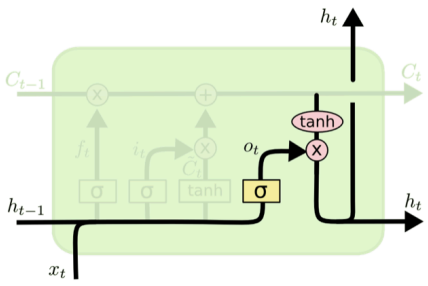


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Figura 8 - Representação e Fórmula Atualização state cell [21]

Definição do output

A ultima etapa é definir o que será a saída. Essa saída é baseada no cell state mas será uma versão filtrada dele. Primeiro é executada a função sigmoide para decidir quais partes do cell state teremos como saída. Depois, passamos o cell state pela função tangente hiperbólica para que os valores fiquem entre -1 e 1 e multiplicamos esses valores pela saída da função sigmoide previamente executada.



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

Figura 9 - Representação e Fórmula output [21]

Aplicações

Já citamos aqui a aplicação de tradução e ela de fato tem um desempenho muito melhor quando usamos LSTM comparado as RNN tradicionais. Esse desempenho melhor não é verificado somente nesse caso. Em

geral quase todas rede neurais recorrentes atualmente utilizam LSTM já que essa memória interna que ele possui é, em geral, extremamente útil [21].

3. Previsão de jogos de futebol

Nesta seção descreveremos a forma como modelamos o problema de prever o resultado de um jogo de futebol.

3.1 Objetivo da previsão

A previsão a ser realizada é se o resultado de um jogo será vitória do mandante, empate ou vitória do visitante. Ou seja, é um problema de classificação com 3 classes possíveis.

3.2 Features

As features que escolhemos usar para nossas previsões foram:

- Quantidade de pontos que os times tem no campeonato atual. Esses são os pontos que as equipes que compõem uma partida conquistaram até agora no campeonato. Cada vitoria no campeonato vale 3 pontos para uma equipe, enquanto que um empate vale 1 ponto e uma derrota não vale nenhum ponto Essa feature é útil para saber o quão bem ou mal uma equipe está naquele campeonato.

- Quantidade de pontos que os times fizeram no ano anterior. Caso uma equipe não tenha jogado o campeonato no ano anterior (pois estava numa divisão inferior) esse campo aparece zerado para essa equipe. Essa feature foi adicionada especialmente para ajudar na previsão dos primeiros jogos do campeonato de um ano, já que a quantidade de pontos disputada naquele ano é ainda muito baixa.

- Quantidade de pontos que o time mandante da partida fez jogando em casa no campeonato atual e quantidade de pontos que o time visitante da partida fez jogando fora de casa no campeonato atual. Essa feature ajuda a determinar o quanto o fator de jogar em casa ou fora de casa faz diferença para uma determinada equipe.

- Quantidade de títulos que as equipes de uma partida venceram nos últimos 40 anos naquele campeonato. Essa feature tenta medir a tradição dessas equipes e o quanto isso pode pesar para o resultado de uma partida.

- Momento atual de cada time. O momento atual indica o quão bem ou mal uma equipe está, qual a fase atual que ela se encontra. Temos duas features de momento, uma calculada com base na sequência de vitórias ou derrotas que uma equipe se encontra. Vale destacar que empates não quebram essa sequência. Se uma equipe venceu 4 jogos, empatou 1 e depois venceu mais 3, ela terá um +7 nessa feature. A segunda feature de momento é calculada apenas nos últimos 5 jogos, onde uma vitória vale +1, uma derrota vale -1 e um empate vale 0. Usando o mesmo exemplo anterior, nessa segunda feature a equipe teria um +4.

- Confronto direto entre as equipes naquele campeonato. Essa feature é calculada com base nos jogos que as equipes já fizeram entre si naquele campeonato. Se uma equipe tem uma pontuação de -5 nessa feature significa que, no período de tempo estipulado, ela perdeu 5 jogos a mais do que ganhou contra a outra equipe. Essa feature tenta medir o quanto repetidas vitórias ou derrotas anteriores contra uma equipe afetam o próximo jogo entre elas.

3.3 Modelos Individuais x Modelos Universais

Para prever o resultado das partidas criamos dois tipos de modelos. A distinção entre modelos individuais e universais que aparece abaixo é apenas em virtude da forma como os modelos são treinados. Nos modelos individuais, primeiro fixamos uma equipe, e um modelo é treinado apenas para prever os jogos dessa equipe. Portanto, os dados de treinamento pertencem somente à essa equipe. Esse modelo aprende apenas como esse time se comporta e se quisermos fazer previsões para outra equipe temos que criar um outro modelo individual separado para essa outra equipe. Nos modelos universais os dados de treinamento vem de vários ou de todos os times. Esses modelos buscam aprender como times de futebol se comportam em geral e não se adaptam a um time específico. Em particular, um modelo universal pode ser utilizado para fazer previsões de qualquer equipe que está presente nos dados de treinamento e, possivelmente até daquelas não presentes nesses dados.

4. Modelos propostos baseados em redes neurais

Nessa seção descrevemos os modelos baseados em redes neurais que propomos para realizar o problema de previsão na seção anterior.

4.1 Modelo Básico

Nesse modelo queremos prever os resultados futuros de uma ou mais equipes. Para isso, entram no modelo as features de jogos anteriores descritas na Seção 3.2. A figura 10 mostra a topologia da rede neural construída. Ela possui duas camadas internas com oito neurônios cada. A camada de entrada tem 13 neurônios (features) e a de saída possui 3 neurônios (os 3 possíveis resultados para um jogo). A função de ativação usada em todas as camadas foi a rectifier (ver seção 2.2.3), com exceção da camada de output onde foi usada a função softmax. Essa função foi usada para obter a probabilidade do resultado pertencer à cada uma das 3 possíveis classes [22]. Além disso, por se tratar de um problema de multi classificação foi usada a função categorical_crossentropy para medir a perda (loss) do modelo [23]. Essa métrica é utilizada durante o treinamento da rede.

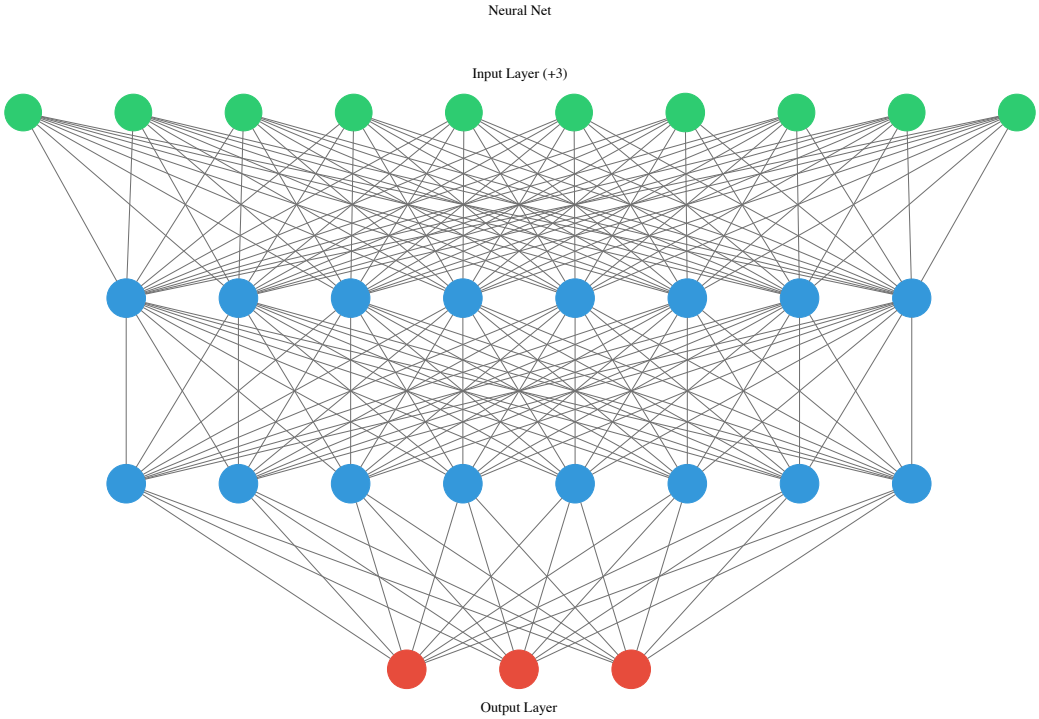


Figura 10 - Topologia da Rede Neural do Modelo Básico

4.2 Modelo LSTM

Neste modelo queremos prever o resultado de uma equipe ou mais equipes em uma rodada. Neste modelo aparece o conceito de rodada já que os modelos LSTM são baseados em séries temporais. A entrada para este modelo é a série dos últimos 5 jogos de uma equipe. Esse modelo busca encontrar o momento que a equipe se encontra baseado nessa série temporal para fazer suas previsões. O modelo pode ser utilizado de forma individual com essa série

de jogos sendo apenas de 1 equipe ou de forma universal usando múltiplas séries temporais, cada uma para uma equipe. Para efeitos de simplificação, apresentaremos aqui o formato dessa entrada para uma equipe apenas (para múltiplas equipes basta concatenar a entrada de cada equipe).

A entrada do modelo está baseado no conceito de janelas. As janelas são uma parte da série temporal e elas vão se deslocando até chegar ao final da série. Como estamos utilizando os últimos 5 jogos de uma equipe, cada janela têm tamanho 5. A primeira janela será das rodadas 1 até 5, a segunda da rodada 2 até 6 e assim por diante (ver Figura 11). Para cada rodada, em cada janela, o modelo recebe todos os resultados da rodada anterior (não só os resultados do time que estamos treinando mas de todos os jogos do campeonato) e qual será a próxima partida do time que estamos treinando.O formato da entrada, portanto, segue a forma tridimensional (janelas, rodadas, resultados da rodada anterior + próxima partida do time X).

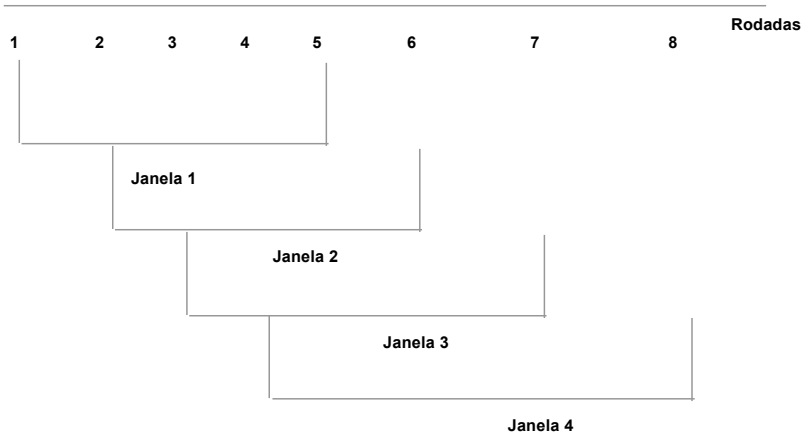


Figura 11 - Representação de janelas

Além da camada LSTM esse modelo possui também uma camada Dropout. Dropout é uma técnica de regularização proposta por Srivastava [24]. Nessa técnica alguns neurônios (escolhidos de forma randômica) são ignorados durante o treinamento. Isso significa que a contribuição que els possuem para ativação de outros neurônios ou para atualização de pesos na rede é temporariamente removida. Quando isso ocorre, outros neurônios terão que assumir o papel dos que foram removidos e acredita-se que isso faz com que a rede neural aprenda representações internas antes desconhecidas. O efeito que isso gera na rede é que ela se torna menos sensível à pesos específicos e mais capaz de generalizar, reduzindo o problema de overfitting que foi notado durante o treinamento dessa rede sem a camada Dropout. A saída da camada LSTM tem

o mesmo número de valores quantos forem os neurônios da rede. Se o LSTM tem 10 neurônios internos então a camada de saída dele também terá 10 neurônios. Para conseguirmos chegar no resultado de classificação com 3 classes (3 neurônios na camada de saída) utilizamos uma camada Dense que faz essa transformação.

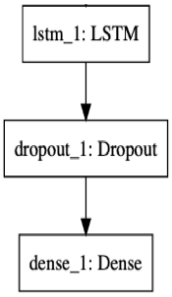


Figura 12 - Modelo LSTM

4.3 Modelo híbrido LSTM

Esse modelo é uma junção do modelo LSTM com o modelo básico. Ele pode ser usado tanto para previsão de resultados para um time ou para várias equipes (“universal”). O modelo está baseado em 2 entradas. A primeira entrada é similar a entrada para o modelo LSTM, excluindo apenas a próxima partida de X, ou seja, está no formato (janelas, rodadas, resultados rodada anterior). A segunda entrada são as features que foram utilizadas nos modelos básicos referentes à equipe que estamos tentando prever e sua adversária naquela rodada. Os resultados obtidos pela camada LSTM do modelo (o “momento” da equipe) são concatenados com as features da segunda entrada. Essa concatenação passa por algumas camadas dense que são redes neurais simples até chegar à previsão do resultado. Além disso, o modelo possui uma camada de dropout entre cada uma das camadas. Novamente, essas camadas foram adicionados para diminuir o problema de overfitting. A figura 13 mostra todas as camadas que este modelo possui.

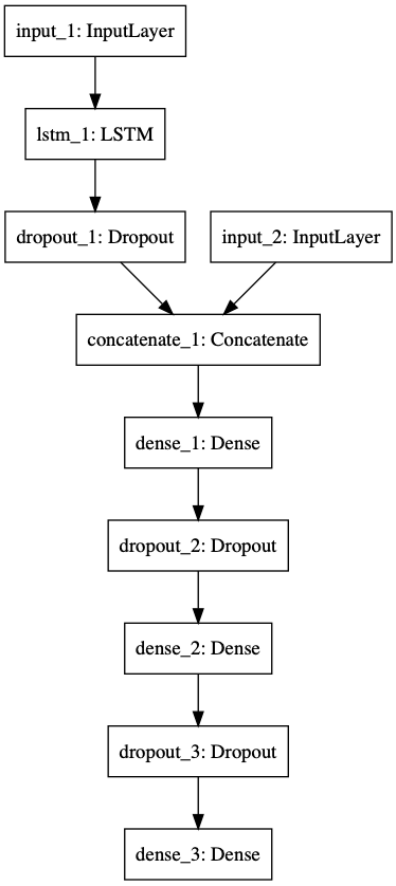


Figura 13 - Modelo híbrido LSTM

5. Resultados dos experimentos

5.1 Ambiente Computacional

Os modelos forma desenvolvidos em Python 3.6 usando as bibliotecas padrão para aprendizado de máquina da linguagem, dentre as quais, se destacam o Pandas, Numpy e Keras. Os programas serão executados em um Macbook Pro que está na versão Mojave do macOS com um processador de 2,7 GHz Intel Core i5.

5.2 Dataset

O dataset utilizado para os experimentos foi obtido em http://www.football-data.co.uk/data.php. Os dados foram coletados de 2005 até 2019 dos campeonatos inglês, espanhol, alemão e italiano. Neste dataset estão disponíveis os resultados finais de todas as partidas do campeonato além de diversas estatísticas dos jogos. Entre essas estatísticas estão, por exemplo chutes a gol, faltas cometidas e cartões recebidos. Todas as informações do

dataset se referem a equipes inteiras e não especificamente à um ou outro jogador.

Para os nossos testes selecionamos os seguintes times do campeonato espanhol: Barcelona, Real Madrid, Atletico de Madri, Valencia, Sevilla e Espanyol. Estas equipes foram escolhidas por representarem uma mistura entre equipes fortes, médias e fracas e pelo fato de estarem presentes no campeonato em todos os anos de 2005 até 2019 (não foram rebaixadas nenhuma vez durante estes anos). No Dataset existem 38 jogos por equipe para cada ano, totalizando 532 jogos para cada equipe.

As informações listadas na seção de features deste documento foram obtidas a partir de dados deste dataset mas não estavam disponíveis a priori no mesmo, ou seja, foram obtidas computacionalmente através de informações disponíveis no dataset. A única informação obtida manualmente foi a estatística de títulos que um time ganhou naquele campeonato. Essa informação foi obtida através de uma pesquisa e foi colocada como uma variável no programa.

Em todos os resultados que serão descritos, foi feita uma divisão do dataset em trainining, validation e test set. O training set têm tamanho de 60% do dataset (8 anos ou 304 jogos por equipe), o validation set tem tamanho de 20% do dataset (3 anos ou 114 jogos por equipe) e o test set também tem tamanho de 20% do dataset (3 anos ou 114 jogos por equipe). Todos os resultados de acurácia reportados abaixo são em relação ao test set.

5.3 Modelos Tradicionais e Baseline

Compararemos os modelos propostos nesse trabalho descritos na seção anterior com modelos tradicionais bem como com um baseline adicional.

Para entendermos quais algoritmos eram mais adequados para o problema proposto foram realizados testes com vários dos algoritmos de aprendizado de máquina.Em todos os testes realizados foram utilizados os algoritmos Linear SVM, Kernel SVM, Naive Bayes, Árvore de Decisão,Random Forest e Rede Neural. Por se tratar de um problema de classificação com mais de duas classes foi utilizada a estratégia one vs rest para reduzir o problema à um problema de classificação binária [25]. Os dois modelos que sempre apresentaram melhor acurácia para todos os 4 campeonatos testados foram Rede Neural e Random Forest e, dessa forma, decidimos utilizar somente estes dois os modelos nos testes adicionais.

Além disso criamos uma baseline que é um modelo que prevê como resultado de suas partidas futuras o resultado mais comum até o momento para aquela equipe. Então se uma equipe ganha mais do que perde ou empata, todos seus resultados futuros serão previstos como sendo vitórias.

5.4 Modelos Individuais

Nesta seção iremos abordar os testes realizados nos modelos individuais, ou seja, aqueles que tentam prever o resultado de apenas uma equipe na rodada.

5.4.1 Resultados dos Experimentos

Os resultados abaixo foram obtidos no test set de cada um dos modelos. As porcentagens de cada modelo sempre correspondem a acurácia obtida. Acurácia aqui representada ele ter acertada exatamente o resultado da partida (vitória mandante, empate ou vitória visitante)

Baseline

O primeiro teste realizado foi com o modelo de previsão baseline, cujos resultados são ilustrados na tabela abaixo.

Tabela 2 - Resultados Baseline

Barcelona	Real Madrid	Atletico de Madrid	Valencia	Sevilla	Espanyol
72,89%	63,55%	59,81%	44,85%	47,66%	37,38%

Como esperado, percebemos que o resultado varia de acordo com a qualidade da equipe, quanto melhor a equipe mais fácil é prever o resultado. Isso será mantido em vários dos modelos aqui propostos mas na baseline é onde vemos a maior diferença entre a melhor e a pior previsão realizada.

Random Forest

O algoritmo foi treinado usando 1000 árvores de decisão sem restrição de tamanho máximo.

Tabela 3 - Resultados Random Forest Individual

Barcelona	Real Madrid	Atletico de Madrid	Valencia	Sevilla	Espanyol
78,48%	77,22%	68,35%	68,35%	53,16%	64,56%

Aqui já é possível perceber uma melhora significativa na previsão, em especial nas equipes em que o baseline tinha mais dificuldade de previsão.

Rede Neural

Foram executadas 200 epochs para o treinamento da rede. Esse número foi escolhido pois a execução de mais epochs posteriores não resultava mais em aumentos suficientes na acurácia do modelo.A rede neural segue a topologia apresentada na seção 4.1 e abaixo temos a tabela de seus resultados.

Tabela 4 - Resultados Rede Neural Individual

Barcelona	Real Madrid	Atletico de Madrid	Valencia	Sevilla	Espanyol
71,71%	65,65%	67,67%	41,41%	60,60%	64,64%

Em geral os resultados para a rede neural foram inferiores ao do random forest e para o Barcelona foram até inferiores ao baseline. Vale destacar, porém que para equipes mais fracas como o Sevilla e o Espanyol os resultados foram superiores aos do random forest.

Modelo LSTM

Esse é o nosso modelo LSTM descrito na Seção 4.3 O LSTM foi treinando usando 15 units (quantidade de neurônios internos que a rede possui) e a taxa de dropout utilizada foi 0,9 [26].

Tabela 5 - Resultados LSTM individual

Barcelona	Real Madrid	Atletico de Madrid	Valencia	Sevilla	Espanyol
74,25%	62,37%	42,57%	43,56%	45,54%	41,58%

Os resultados para este modelo foram bastante inferiores aos anteriores. Uma possível explicação aqui é que a quantidade de parâmetros passados para o LSTM é muito grande. Na parte de resultados da rodada anterior, como temos 40 equipes que frequentaram o campeonato nos 15 anos, temos 1600 valores de atributos sendo passados para o modelo (representação One-Hot-Encoder do nome das equipes para o resultado [27]).

Modelo Híbrido LSTM

Esse é o nosso modelo LSTM descrito na Seção 4.4. A parte LSTM deste modelo foi construída com os mesmos parâmetros descritos anteriormente, com a adição de kernel regularizers na camada LSTM para reduzir o overfitting que

estava acontecendo neste modelo. Além disso estes mesmo regularizers foram utilizados nas 2 camadas Dense intermediárias.O resultado desse modelo é apresentado na tabela abaixo.

Tabela 6 - Resultados Modelo Híbrido individual

Barcelona	Real Madrid	Atletico de Madrid	Valencia	Sevilla	Espanyol
81,18%	64,35%	64,35%	56,43%	50,49%	41,58%

Neste modelo, apesar de termos os mesmos 1600 valores de atributos descritos no LSTM, tivemos resultados melhores. Isso pode se dever ao fato de que aqui temos um segundo input com as informações listadas na Seção 3.2. Essas informações parecem contribuir para uma melhora significativa do modelo.

Resumo dos Modelos Individuais

Tabela 7- Resumo Resultados Modelos Individuais

	Geral	Barcelona	Real Madrid	Atletico de Madrid	Valencia	Sevilla	Espanyol
Baseline	54,35%	72,89%	63,55%	59,81%	44,85%	47,66%	37,38%
Random Forest	**68,35%**	78,48%	**77,22%**	**68,35%**	**68,35%**	53,16%	64,56%
Rede Neural	61,94%	71,71%	65,65%	67,67%	41,41%	**60,60%**	**64,64%**
LSTM	51,64%	74,25%	62,37%	42,57%	43,56%	45,54%	41,58%
Híbrido LSTM	59,73%	**81,18%**	64,35%	64,35%	56,43%	50,49%	41,58%

Podemos perceber que em geral o modelo que obteve o melhor resultado na previsão individual das equipes foi o random forest. Também é possível notar que o modelo Híbrido LSTM apresentou resultados inferiores aos da Rede Neural. Uma possível explicação para isso é o fato desse modelo utilizar uma grande quantidade de parâmetros e a quantidade de dados que temos para treiná-lo ser relativamente pequena (342 jogos no training set para cada equipe). O uso de camadas de dropout e de kernel regularizers buscou minimizar esse problema mas mesmo assim ele apareceu nos resultados.

5.5 Modelos Universais

Nesta seção iremos abordar os resultados dos experimentos nos modelos universais, ou seja, aqueles que tentam prever o resultado de várias ou todas as equipes.

5.5.1 Resultados dos experimentos

Nesta seção apresentaremos os resultados que obtivemos nos modelos universais. Para cada modelo será dada a acurácia geral do modelo treinado com dados das 6 equipes que estamos trabalhando e uma tabela com a acurácia que ele obteve prevendo cada uma delas.

Rede Neural

A rede neural usada foi a mesma construída para o treinamento individual das equipes e neste modelo universal apresentou acurácia de 73,98% no test set. Abaixo temos os resultados que ele apresentou para cada uma das equipes.

Tabela 8- Resultados Rede Neural Universal

Barcelona	Real Madrid	Atletico de Madrid	Valencia	Sevilla	Espanyol
85,85%	77,77%	76,76%	66,66%	69,69%	67,67%

Random Forest

O random Forest também foi treinado usando 1000 árvores e obteve acurácia de 75,53% no test set que contem as 6 equipes.Abaixo temos os resultados que ele apresentou para cada uma das equipes.

Tabela 9 - Resultados Random Forest Universal

Barcelona	Real Madrid	Atletico de Madrid	Valencia	Sevilla	Espanyol
83,83%	75,75%	80,80%	67,67%	73,73%	69,69%

LSTM

O LSTM universal também foi treinando com 15 units e taxa de Dropout 0,9. Abaixo temos os resultados que ele apresentou para cada uma das equipes.

Tabela 10- Resultados LSTM Universal

Barcelona	Real Madrid	Atletico de Madrid	Valencia	Sevilla	Espanyol
71,28%	58,41%	58,41%	44,55%	46,53%	37,40%

O modelo LSTM foi o único que teve resultados piores na forma universal comparado à forma individual. Nos dois casos, os resultados foram bastante inferiores aos outros modelos propostos. Novamente, aqui isso pode ser pelo quantidade muito grande de parâmetros passados para ele. Em especial, no modelo universal, a quantidade de parâmetros aumenta ainda mais já que estamos passando uma concatenação que é seis vezes maior do que o que estávamos passando no modelo individual.

Modelo Híbrido LSTM

O modelo híbrido LSTM foi treinado com 400 epochs e obteve acurácia de 72,43% no test set para as 6 equipes. Quando quebramos os resultados desse modelo para ver o quanto ele acertou para cada equipe temos os seguintes resultados.

Tabela 11 - Resultados Modelo Híbrido Universal

Barcelona	Real Madrid	Atletico de Madrid	Valencia	Sevilla	Espanyol
84,15%	68,31%	77,22%	68,31%	63,36%	62,37%

Aqui é interessante notar que apesar do aumento do número de parâmetros, o modelo melhorou significativamente quando comparado a sua versão individual.

Resumo dos Modelos Universais

Tabela 12 - Resumo dos Resultados Modelos Universais

	Geral	Barcelona	Real Madrid	Atletico de Madrid	Valencia	Sevilla	Espanyol
Random Forest	73,98%	**85,85%**	**77,77%**	76,76%	66,66%	69,69%	67,67%
Rede Neural	**75,53%**	83,83%	75,75%	**80,80%**	67,67%	**73,73%**	**69,69%**
LSTM	56,60%	71,28%	58,41%	58,41%	44,55%	46,53%	37,40%
Híbrido LSTM	72,43%	84,15%	68,31%	77,22%	**68,31%**	63,36%	62,37%

Dentre os modelos universais, a rede neural foi a que apresentou melhor acurácia geral. Podemos perceber que agora, com uma quantidade de dados para treino maior (2052 jogos no total) o modelo híbrido LSTM apresentou

resultados superiores à sua versão individual mas ainda não conseguiu superar a rede neural comum. Novamente, isso provavelmente ocorre pela falta de dados para treinar o grande número de parâmetros que esse modelo possui.

5.6 Modelos universais: Previsão de time fora dos exemplos de treino

Para testar a capacidade de generalização dos modelos universais, utilizamos os modelos universais treinados na seção anterior para prever um outro time (Atlético de Bilbao) que não foi visto durante o treinamento. Comparamos a qualidade dessas previsões com a previsão dos modelos individuais treinados exclusivamente para prever o Atlético de Bilbao.

Tabela 13 - Resultados Modelos Atlético de Bilbao

	Modelo Individual	Modelo Universal
Random Forest	**63,64%**	**69,70%**
Rede Neural	62,62%	65,65%
LSTM	37,62%	41,58%
Híbrido LSTM	50,50%	60,39%

5.7 Discussão dos Resultados

Podemos perceber que os modelos universais obtiveram desempenho bastante superior aos modelos individuais. Isso pode ser explicado pois nos modelos universais existe um contexto muito maior de informações em que o modelo pode basear as suas informações. Ao invés de termos uma visão fixada sobre apenas uma equipe temos um espectro amplo que permite ao modelo aprender informações sobre várias equipes e em última instância, sobre o campeonato em geral. Também é possível notar que tantos nos modelos individuais como nos universais, a previsão é bastante afetada por qual equipe se está tentando prever. É muito mais fácil para todos os modelos prever equipes que costumam ter resultados constantes como por exemplo o Barcelona.

Uma das maiores dificuldades é a questão que já foi levantada de não serem sempre as mesmas equipes no campeonato. Nos 15 anos de dados que foram obtidos só 7 equipes se mantiveram na primeira divisão o tempo inteiro. Isso acaba dificultando métricas como pontuação no ano anterior (que é zerada

quando a equipe não esteve no campeonato no ano anterior) e head to head que ficam com um espaço amostral menor.

Em termos específicos de cada modelo percebemos que em média os modelos random forest e rede neural tiveram desempenho melhor para essas seis equipes que estamos tentando prever. O modelo híbrido LSTM foi muito bom para prever resultados do Barcelona mas teve desempenho inferior na previsão das outras equipes. Mesmo assim, esse modelo apresentou um comportamento consideravelmente superior quando ele foi treinado de forma universal (com a concatenação dos dados das seis equipes como entrada) em comparação à quando ele foi treinado de forma individual com os dados de apenas uma equipe. Também é interessante notar que a adição de uma segunda entrada e de camadas Dense nesse modelo melhorou muito seu desempenho quando comparado ao modelo LSTM. Essas adições permitiram que o modelo lidasse melhor com a grande quantidade de parâmetros que ele possui.

6. Considerações finais

6.1 Conclusões

No âmbito pessoal, o projeto como um todo foi bastante útil já que consegui aprender muito mais sobre algoritmos de aprendizado de máquina e aplicá-los a um problema que possuo grande interessante. O objetivo que tínhamos no projeto era exatamente ter um maior entendimento sobre essa área e isso foi cumprido. Mesmo já conhecendo a linguagem de programação consegui testar e utilizar muitas bibliotecas e algoritmos que me eram até então desconhecidos. Além disso consegui ter um entendimento maior dos problemas e dificuldades que existem na previsão de jogos de futebol e sem dúvida é algo que irei pesquisar ainda mais a fundo no futuro. Em termos de resultados, alguns de fato foram surpreendentes como o desempenho ruim do LSTM mas ter aprendido e visto as aplicações deste algoritmo foi bastante interessante. A forma de aprendizado, utilizando um curso online e tendo reuniões periódicas com o orientador também foi uma excelente parte do projeto.

6.2 Trabalhos Futuros

A área de previsões de jogos de futebol contêm muitas possibilidades. Algo que não estava no escopo desse projeto mas acredito que seria interessante para a academia seria tentar combinar as features que utilizamos aqui que eram sobre equipes como um todo, com features individuais sobre jogadores para tentar entender a importância de cada jogador para uma partida.

Além disso considerar a construção de um banco de dados bem modelado para o problema pode ser interessante como forma de ter-se um repositório central com todas as estatísticas das partidas e dos jogadores.

7. Referências

[1] Brian Macdonald. An Expected Goals Model for Evaluating NHL Teams and Players. **MIT Sloan Sports Analytics Conference**, 2012, 8p. Disponível em: http://www.sloansportsconference.com/wp-content/uploads/2012/02/NHL-Expected-Goals-Brian-Macdonald.pdf . Acesso em: 14 out. 2019.

[2] Patrick Lucey, Alina Bialkowski, Mathew Monfort, Peter Carr, Iain Matthews. Quality vs Quantity: Improved Shot Prediction in Soccer using Strategic Features from Spatiotemporal Data. **MIT Sloan Sports Analytics Conference**, 2015, 11p. Disponível em: http://www.sloansportsconference.com/wp-content/uploads/2015/02/SSAC15-RP-Finalist-Quality-vs-Quantity.pdf. Acesso em: 14 set. 2019.

[3] Harm Eggels, Ruud van Elk, Mykola Pechenizkiy. Explaining soccer match out- comes with goal scoring opportunities predictive analytics. **Eindhoven University of Technology**, 2016, 10p. Disponível em: https://dtai.cs.kuleuven.be/events/MLSA16/papers/paper_16.pdf. Acesso em: 20 jun. 2019.

[4] Boice, Jay. How Our Club Soccer Predictions Work, 2019. Disponível em: https://fivethirtyeight.com/methodology/how-our-club-soccer-predictions-work/, Acesso em: 14 out. 2019

[5] A.E. Elo. **The rating of chess players, past and present.** Arco Publishing, 1978. 220p.

[6] Lars Magnus Hvattuma, Halvard Arntzen. Using ELO ratings for match result prediction in association football. **International Journal of Forecasting 26**, 2010 p.460-470. Disponível em : http://www.collective-behavior.com/publ/ELO.pdf . Acesso em : 20 ago. 2019.

[7] Anthony Costa Constantinou, Norman Elliott Fenton. Determining the level of ability of football teams by dynamic ratings based on the relative discrepancies in scores between adversaries. **Journal of Quantitative Analysis in Sports,** 2013.

[8] Herbinet, Corentin, Predicting Football Results Using Machine Learning Techniques, 2018.Disponível em: https://www.imperial.ac.uk/media/imperial-college/faculty-of-engineering/computing/public/1718-ug-projects/Corentin-Herbinet-Using-Machine-Learning-techniques-to-predict-the-outcome-of-profressional-football-matches.pdf. Acesso em: 30 jun 2019.

[9] Football Data Team, Historical Football Results and Betting Odds Data, 2019. Disponível em : http://www.football-data.co.uk/data.php. Acesso em: 30 mar. 2019.

[10] Football API Team, API Football, 2019. Disponível em: https://www.api-football.com. Acesso em: 30 mar. 2019.

[11] Injuries and Suspensions Team, Injuries and Suspensions, 2019. Disponível em: https://injuriesandsuspensions.com. Acesso em: 30 mar.2019.

[12] Who Scored Team, Football Statistics, 2019 Disponível em:https://www.whoscored.com/Statistics. Acesso em: 30 mar.2019.

[13] Emerenko, Kirill, Machine Learning A-Z™: Hands-On Python & R In Data Science,2018. Disponível em: https://www.udemy.com/machinelearning/. Acesso em 15 mar.2019.

[14] Rokach Lior, Maimon Oded. Data Mining and Knowledge Discovery Handbook, 2nd edition. Chapter 9: Decision Trees. p.165-192 ,2010.

[15] Peng, Wei, Juhua Chen, and Haiping Zhou. An implementation of ID3-decision tree learning algorithm. 2009, 22p.Disponível em: https://rhuang.cis.k.hosei.ac.jp/Miccl/AI-2/L10-src/DecisionTree2.pdf. Acesso em: 20 nov. 2019

[16] Liaw, Andrew. Wiener, Matthew. Classification and Regression by randomForest, 2002.Disponível em: https://www.researchgate.net/profile/Andy_Liaw/publication/228451484_Classification_and_Regression_by_RandomForest/links/53fb24cc0cf20a45497047ab/Classification-and-Regression-by-RandomForest.pdf. Acesso em: 20 set. 2019.

[17] Shotton, Jamie, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from single depth images,2011.Disponível em: https://

www.microsoft.com/en-us/research/wp-content/uploads/2016/02/BodyPartRecognition.pdf. Acesso em: 14 set. 2019.

[18] Widrow, Bernard, E. Rumelhart, David, A.Lehr, Michael, Neural networks: applications in industry, business and science. 1994

[19] Bottou, Léon. Stochastic Gradient Learning in Neural Networks, 1991. Disponível em: https://leon.bottou.org/publications/pdf/nimes-1991.pdf. Acesso em: 20 nov.2019.

[20] Karpathy, Andrej. The Unreasonable Effectiveness of Recurrent Neural Networks,2015. Disponível em: http://karpathy.github.io/2015/05/21/rnn-effectiveness/. Acesso em: 20 set. 2019.

[21]Colah. Understanding LSTM Networks, 2015. Disponível em: https://colah.github.io/posts/2015-08-Understanding-LSTMs. Acesso em: 22 set.2019

[22] Lan, Haihan. The Softmax Function, Neural Net Outputs as Probabilities, and Ensemble Classifiers,2017. Disponível em: https://towardsdatascience.com/the-softmax-function-neural-net-outputs-as-probabilities-and-ensemble-classifiers-9bd94d75932. Acesso em: 24 set.2019

[23] Developers, Google. Descending into ML: Training and Loss, 2019 Disponível em: https://developers.google.com/machine-learning/crash-course/descending-into-ml/training-and-loss. Acesso em 26 nov. 2019.

[24] Nitish Srivastava, Geoffrey Hinton,Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, Dropout: A Simple Way to Prevent Neural Networks from Overfitting,2014. Disponível em: http://jmlr.org/papers/v15/srivastava14a.html. Acesso em: 24 out. 2019

[25] scikit-learn developers, sklearn.multiclass.OneVsRestClassifier, 2019 Disponível em: https://scikit-learn.org/stable/modules/generated/sklearn.multiclass.OneVsRestClassifier.html. Acesso em: 23 mar.2019.

[26] Brownlee, Jason. How to Develop LSTM models for Time Series Forecasting,2018. Disponível em: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/. Acesso em: 20 jun.2019.

[27] scikit-learn developers, sklearn.preprocessing.OneHotEncoder, 2019, http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html . Acesso em: 23 mar.2019