Mineração de Dados no Contexto do Futebol Americano

Lucas de Almeida Carvalho[1], Jean Paul Barddal[2]

[1]Escola Politécnica - Pontifícia Universidade Católica do Paraná (PUCPR) – Curitiba, PR – Brasil

[2]PPGIA – Programa de Pós Graduação em Informática Aplicada - Pontifícia Universidade Católica do Paraná (PUCPR) – Curitiba, PR – Brasil

lucas130421@gmail.com, jean.barddal@ppgia.pucpr.br

Abstract. *The article presents the use of machine learning to predict the number of yards gained per play in the National Football League (NFL). The author uses historical data from NFL games to develop prediction models that take into account various variables, such as field position, distance to the goal line, remaining time in the game, and team formation. The results showed that the machine learning models were able to predict the number of yards gained per play with high accuracy, which could have significant implications for football teams, allowing them to optimize their game strategies and improve their on-field performance. The article concludes by highlighting the potential of machine learning to transform sports data analysis and improve decision-making in professional sports competitions.*

Resumo. *O artigo apresenta o uso de machine learning para prever o número de jardas ganhas por jogadas na National Football League (NFL). O autor utiliza dados históricos de jogos da NFL para desenvolver modelos de previsão que levam em conta várias variáveis, como a posição de campo, a distância para a linha de gol, o tempo restante do jogo e a formação tática das equipes. Os resultados mostraram que os modelos de machine learning foram capazes de prever com alta precisão o número de jardas ganhas por jogadas, o que pode ter implicações significativas para as equipes de futebol americano, permitindo que elas otimizem suas estratégias de jogo e melhorem seu desempenho em campo. O artigo conclui destacando o potencial do machine learning para transformar a análise de dados esportivos e melhorar a tomada de decisões nas competições esportivas profissionais.*

1. Introdução

O futebol americano é um esporte que se baseia em estratégias, jogadas e desempenho individual e coletivo dos jogadores, o grande objetivo de um time é a conquista de território adversário até chegar a tão esperada End Zone e marcar um touchdown. Nesse contexto, a quantidade de jardas que uma equipe é capaz de avançar em uma jogada é uma estatística de grande importância. O ganho de jardas pode fazer a diferença entre manter a posse de bola, marcar um touchdown ou ser forçado a chutar a bola para o adversário, no entanto, prever o ganho de jardas em uma jogada não é uma tarefa fácil, pois depende de diversos fatores, como a formação da equipe adversária, a habilidade do

corredor, a precisão do passe, entre outros; e para tal, ter um playbook eficiente é uma das chaves para o sucesso de um time.

O playbook é uma das ferramentas mais importantes para uma equipe de futebol americano pois contém as jogadas que serão utilizadas em diferentes situações do jogo. Para que uma equipe tenha sucesso, é necessário que o playbook contenha jogadas que retornem bons resultados e que sejam eficazes contra diferentes adversários. O ganho de jardas é uma métrica fundamental para avaliar o desempenho dessas jogadas.

Para atingir esse objetivo, técnicas de machine learning podem ser aplicadas ao processo de criação do playbook. A partir da análise de dados históricos, como estatísticas de jogadas passadas e formações de equipes, é possível identificar as jogadas que tiveram maior sucesso em diferentes situações de jogo e adaptá-las para melhor performance.

Neste artigo, exploramos como as técnicas de machine learning podem ser aplicadas na previsão de ganho de jardas em jogadas de futebol americano. Discutiremos como o pré-processamento de dados e a seleção de características podem ser usados para melhorar a precisão do modelo. Além disso, abordaremos modelos como o XGBoost, LightGBM e Linear Regression e como estes podem ser eficazes na previsão de valores para atributos numéricos, como o número de jardas.

2. Metodologia

O dataset utilizado neste artigo faz parte do desafio de 2020 do projeto Big Data Bowl. O Big Data Bowl é um projeto da NFL em parceria com a Amazon Web Services (AWS) que tem como objetivo incentivar a análise avançada de dados esportivos e descobrir insights valiosos para a liga de futebol americano. O projeto anualmente disponibiliza um conjunto de dados para os participantes analisarem e apresentarem suas descobertas e modelos preditivos.[1]

Cada edição do Big Data Bowl começa com a divulgação de um conjunto de dados de jogos da temporada regular da NFL e um problema a ser abordado. Os participantes têm algumas semanas para analisar os dados e criar seus modelos, e em seguida apresentam suas descobertas para uma banca de jurados compostos por especialistas em estatística, ciência de dados e NFL.

Os vencedores do Big Data Bowl recebem prêmios em dinheiro e podem ter a oportunidade de se envolver em projetos de análise de dados com a NFL ou outras organizações esportivas. Além disso, a liga utiliza as descobertas e modelos gerados pelos participantes do Big Data Bowl para aprimorar suas estratégias e tomadas de decisão em relação ao desempenho dos jogadores e equipes.

O dataset utilizado é composto por dados das temporadas 2017, 2018 e 2019 da liga profissional de futebol americano, a NFL. Nele podemos encontrar informações sobre

[1] Desafio pode ser acessado através do link https://www.kaggle.com/competitions/nfl-big-data-bowl-2020

times envolvidos na partida, formação ofensiva de cada jogada, posisionamento dos jogadores em campo, direção das jogadas, condições climaticas durante o jogo como também resultados de cada jogada, ou seja, a quantidade de jardas que foram ganhas ou perdidas, dentre outras informações.

Neste dataset são totalizados 682.154 linhas, sendo cada uma delas uma jogada diferente e o arquivo pode ser encontrado no link **https://www.kaggle.com/competitions/nfl-big-data-bowl-2020.**

A ideia é aplicar diversas técnicas de pré-processamento dos dados usando bibliotecas python como Pandas, Numpy, dentre outras e na sequência, submeter o dataset como entrada de treino em um modelo regressor, o qual fará a predição das jardas que esta jogada poderá retonar.

Abaixo, na Figura 1, podemos encontrar a distribuição dos valores do atributo alvo, Yards, o qual são as jardas a serem previstas.

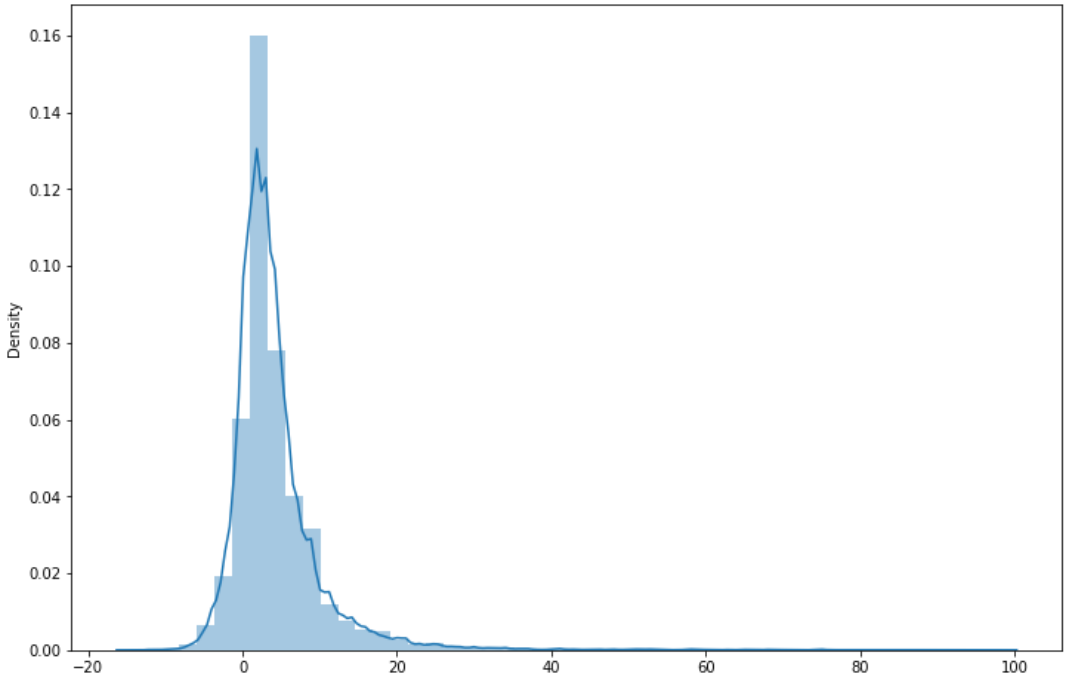

Figura 1 – Distribuição da variável alvo do dataset – Yards.

E logo abaixo, na Tabela 1, temos uma relação dos principais atributos presentes no dataset, com destaque para a coluna Yards, atributo qual será o objetivo de predição deste projeto.

Atributo	Descrição
X	Posição do jogador no eixo X do campo

Y	Posição do jogador no eixo Y do campo
Game Clock	Tempo restante de jogo
S	Velocidade do recebedor *(Yards/seconds)*
A	Velocidade de aceleração do recebedor *(Yards/Seconds)*
Quarter	Quarto atual durante a jogada em questão
Season	Temporada da competição
Time Handoff	Tempo de release da bola pelo Quarterback
Possession Team	Código do time com a posse de bola
NflIdRusher	Id da NFL do jogador que recebeu a bola
Orientation	Orientação do jogador em relação ao campo *(Em Graus°)*
Dir	Ângulo de movimento da jogada *(Em Graus°)*
Temperature	Temperatura no momento da jogada *(° Fahrenheit)*
Field Position	Indicativo se a jogada ocorre no campo adversário ou não
HomeScoreBeforePlay	Placar do time da casa antes da jogada acontecer
Yards	Resultado da jogada em jardas obtidas com a jogada.

Tabela 1 – Relação das principais variáveis do dataset.

Como partes do processo deste projeto, teremos a etapa de coleta dos dados, que é a obtenção do dataset no Kaggle, posteriormente, partimos para o pré-processamento dos dados, que é onde aplicamos técnicas como limpeza, normalização, análise exploratória dos dados presentes no dataset, dentre outras técnicas deixando-o preparado para entrada no modelo de regressão, após esta etapa, utilizaremos um subset do conjunto original para treinamento de 3 modelos distintos, realizando tuning de parâmetros e, se necessário, novas etapas de pré-processamento, sempre ao final de cada etapa, realizando a avaliação das métricas escolhidas afim de encontrar os melhores resultados para o problema descrito.

Abaixo, na Figura 2, encontramos um resumo dos processos efetuados para o desenvolvimento deste projeto.

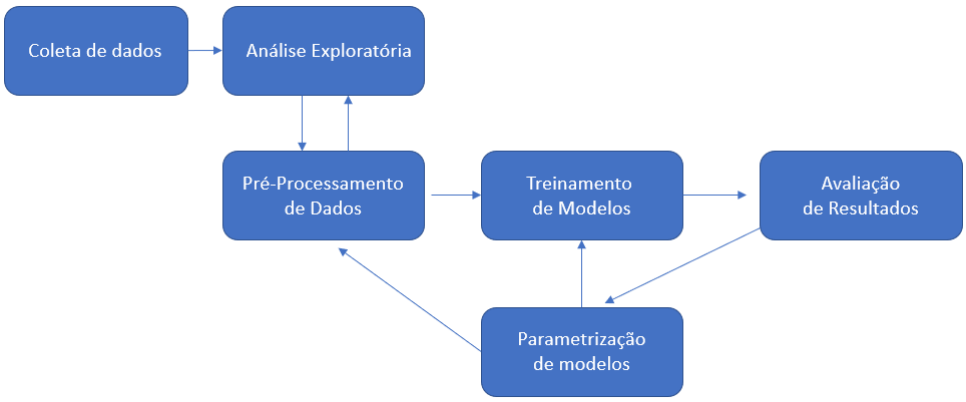

Figura 2 – Resumo do processo aplicado na execução do projeto.

Como alternativas para treinamento foram selecionados os modelos LinearRegression, XGBoost e LightGBM. O modelo LinearRegression foi escolhido devido a ser um modelo conhecido e ter uma facilidade de explicar a lógica por trás de cada previsão. Já o modelo LightGBM foi escolhido por ter entregado bons resultados em outras aplicações desenvolvidas anteriormente e houve a expectativa de que entregaria bons resultados neste cenário também. Já o modelo XGBoost foi escolhido devido a sua recente fama em performar muito bem em problemas de regressão, por este motivo, foi considerado como uma alternativa de modelo.

Como avaliação dos resultados, as métricas de Root Mean Squared Error (RMSE), R2 Score e Mean Absolute Error (MAE), foram escolhidas para mensurar a qualidade dos modelos, visto que neste cenário, precisamos manter uma margem de erros bem próxima de 0, pois é o que vai determinar o sucesso de uma jogada ou não, abaixo encontramos uma breve explicação sobre cada métrica.

- **R2 score**: Mede a proporção da variabilidade na variável dependente explicada pelas variáveis independentes e varia de 0 a 1. Um valor de 1 indica um ajuste perfeito do modelo aos dados, enquanto um valor de 0 indica que o modelo não explica nenhuma variabilidade. Um R2 score alto é considerado um bom ajuste do modelo.
- **RMSE:** mede a diferença entre os valores previstos e observados da variável dependente, sendo calculado como a raiz quadrada da média dos erros quadrados. É útil para identificar valores discrepantes e quanto menor seu valor, maior é a precisão do modelo.
- **MAE:** mede a magnitude média dos erros de previsão do modelo em relação aos valores observados da variável dependente, sendo calculado como a média dos valores absolutos dos erros. É uma medida robusta, não afetada por valores discrepantes e quanto menor seu valor, maior é a precisão do modelo.

3. Execução do Projeto

3.1 Coleta dos dados

Os passos efetuados para a coleta dos dados foram o download do arquivo *.csv* do repositório Kaggle e posteriormente a adição deste em um diretório pessoal do google drive, o qual foi importado para dentro do ambiente do Google Colab através da biblioteca Drive do próprio Google Colab, em seguida, os dados foram carregados para um dataframe usando a biblioteca Pandas.

3.2 Pré-Processamento dos Dados

Uma das fases mais importantes de um projeto de ciência de dados é o pré-processamento dos dados, pois para obter bons resultados, como uma margem de erro próxima de 0, objetivo principal deste projeto, precisamos que os atributos de entrada de treinamento dos modelos expliquem ao máximo possível o cenário do problema.

Para o dataset atual, temos diversas colunas que estão totalmente desnormalizadas e com valores extremamente oscilantes, antes de iniciar o pré-processamento de cada uma delas, uma nova coluna foi adicionada ao conjunto, sendo esta o resultado da jogada. Com base na quantidade de jardas ganhas e a distância inicial da jogada, podemos dizer se ela resultou em First Down, Turn Over on Downs, perdas de jardas ou se a posição de campo continua a mesma, ou seja, não houve avanço e nem recuo. Após a adição do coluna supracitada, é possível validar o desempenho dos times em cada temporada logo abaixo na figura 3.

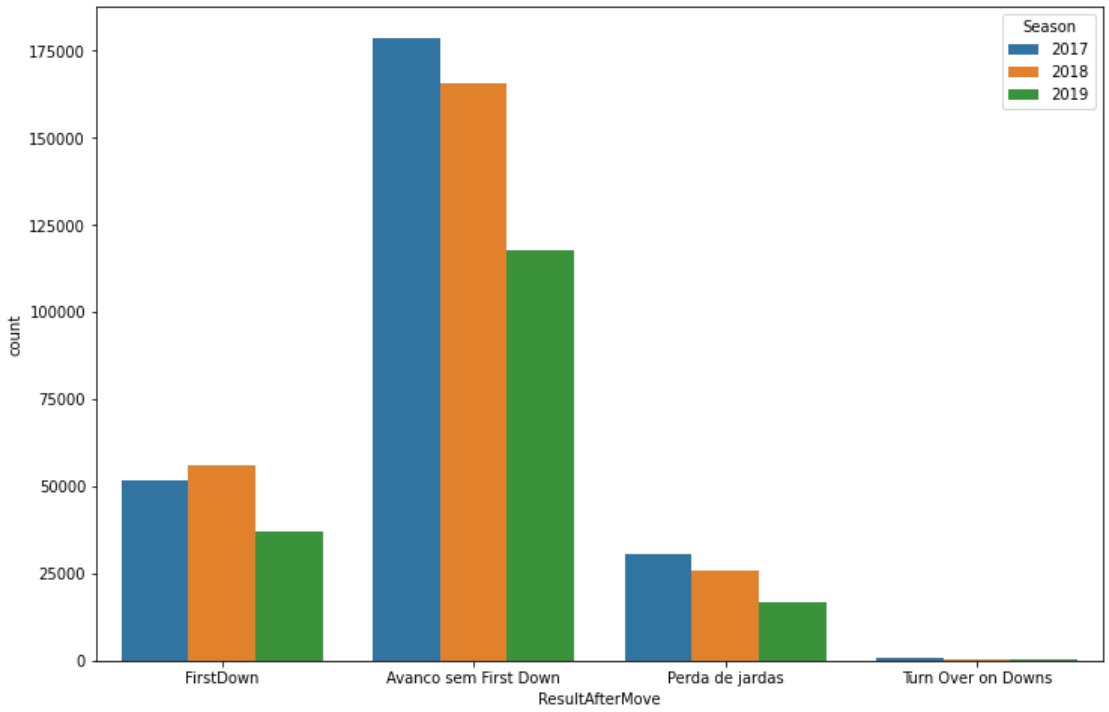

Figura 3 – Distribuição de cada resultado de jogadas por temporada

Após esta validação, o dataset foi dividido entre dois subsets, sendo X, os dados a serem utilizados para abstração do problema e Y a variável alvo que queremos prever, as jardas. Logo em seguida, dividimos novamente o dataset utilizando o método *Train_Test_Split()* da biblioteca Sklearn.model_selection, utilizando uma proporção de 70/30 e usando o conjunto Y para divisão dos valores proporcionalmente.

O próximo passo, agora trabalhando primeiramente com os dados de treino, foi validar o percentual de dados nulos em cada coluna, através do método *IsNull()* do pandas, chegando ao valores da imagem 4, logo abaixo.

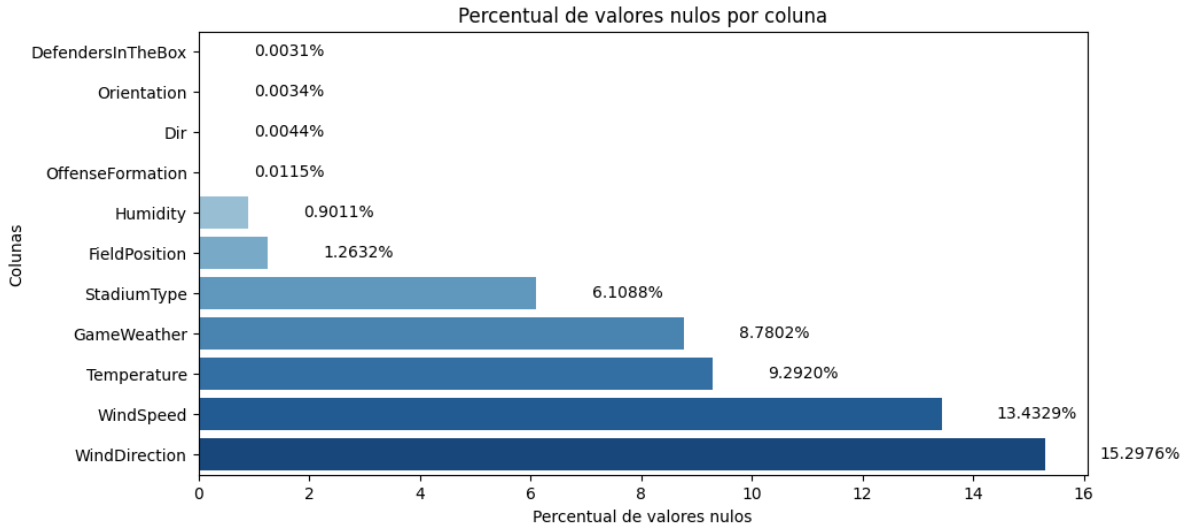

Figura 4 – Percentual de dados nulos por coluna

Em seguida, teremos uma breve descrição sobre a tratativa aplicada às principais colunas.

Para as colunas Orientation e Dir, como fora validado que suas distribuições segue um padrão normal e a quantidade de informações nulas é um valor irrisório, o preenchimento destes foram efetuados utilizando a média das colunas.

Para a coluna fieldPosition, a seguinte lógica foi aplicada: utilizando de uma segunda coluna, a YardLine e partindo do princípio de que um campo de futebol americano é divido entre dois lados, time visitante e time mandante, a linha de 50 jardas trata-se do ponto limite entre estes, neste caso, apenas foi verificado se a posição de campo da jogada era menor ou igual à 50, em caso positivo, a jogada ocorreu no campo do time que esta com a posse de bola, caso contrário, a equipe já estava no campo do time adversário.

Para as colunas Stadium Type, Game Weather e Wind Direction, visto a grande variedade de valores em cada, os dados foram agrupados conforme o evento descrevia situações semelhantes e desta forma chegamos a padronização dos dados, para os valores nulos, 'N/A' foi aplicado, visto que, tratando-se de uma informação especifica, o preenchimento com o método *Mode()*, por exemplo, não traria benefícios.

Para a coluna Wind Speed, foi removido os caracteres de string e os valores foram transformados no tipo Int, nos casos em que apresentava ranges de velocidade (ex: 50-60 mph), houve o retorno da média entre os dois valores e para os casos nulos, a velocidade foi fixada em 0.

Já nas colunas Humidity e Temperature, o método *ffill()* foi utilizado no preenchimento dos valores, uma vez que este tipo de dado tende a ter pouca variação de uma linha para a outra.

Ao final da do pré-processamento de todas as features, foi utilizado um laço *for()* para tentar transformar todos as colunas para o tipo *'Float16'* e uma exception adiciona as colunas que retornam erro, ou seja, que possuem atributos categóricos e não conseguem ser transformados, à uma lista na qual percorremos aplicando o método *LabelEncoder()*

da biblioteca sklearn.preprocessing. Este método aplica a transformação de todas as variáveis categóricas em valores numéricos e sequenciais, onde cada valor identifica uma categoria em determinada coluna.

Após todos as etapas percorridas, temos o dataset pré-processado e começamos o treinamento de modelos.

3.3 Treinamento

3.3.1 – Linear Regression

Como primeira tentativa de treinamento, foi utilizado o modelo Linear Regression da biblioteca sklearn, neste modelo o objetivo é encontrar a equação que melhor descreve a relação entre as variáveis independentes e dependentes. Essa equação é então usada para prever valores da variável dependente com base nos valores das variáveis independentes. A equação é geralmente na forma de Y = a + bX, onde Y é a variável dependente, X é a variável independente, a é o intercepto e b é o coeficiente de regressão. Para o treinamento em questão, o modelo foi utilizado com seus parâmetros em modo default e os dados foram padronizados em uma única escala, utilizando o método *MinMaxScaler()*, também da biblioteca sklearn. O método funciona subtraindo o valor mínimo do conjunto de dados e dividindo pelo intervalo entre o valor mínimo e o máximo, a equação esta representada abaixo na figura 5.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Figura 5 – Equação MinMaxScaler

Na tabela 2 abaixo, temos o resultado das métricas escolhidas para avaliação do modelo e logo em seguida encontramos, na figura 5, o gráfico de resíduos que nos mostra a distribuição dos erros entre os valores originais e os valores previstos.

Modelo	Logistic Regression	
RMSE	R2 Score	MAE
6.39	1.6%	3.78

Tabela 2 – Resultados Linear Regression.

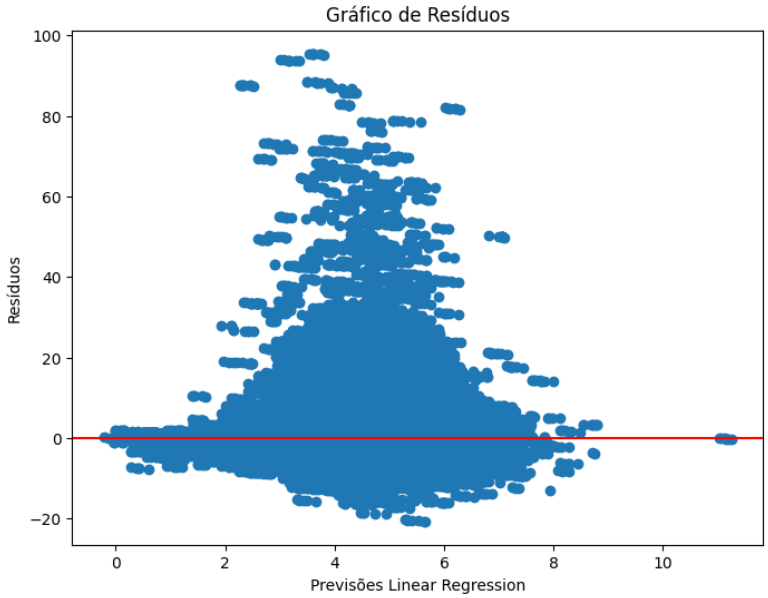

Figura 6 - Gráfico de Resíduos Linear Regression

Analisando as métricas do modelo e também o gráfico de resíduos acima, podemos concluir que a previsão não teve um grande percentual de sucesso, temos uma concentração grande de erros que chegam a quase 100 jardas de diferença entre um valor previsto e um valor real, o que nos leva a crer que, para o problema em questão, não se trata de uma boa alternativa.

3.3.2 LightGBM

Para uma segunda tentativa, o modelo LightGBM foi escolhido, este é um modelo de regressão de gradient boosting que utiliza árvores de decisão para fazer previsões, ele utiliza uma estratégia de treinamento por amostragem, que permite treinar árvores de decisão mais profundas e precisas em cada iteração, em resumo, a cada rodada o modelo tenta corrigir os erros da previsão anterior ajustando uma nova árvore de decisão buscando uma melhor precisão para que o modelo final seja capaz de explicar a maior parte da variação nos dados.

Neste modelo também foi utilizado o método MinMaxScaler, supracitado, para padronizar as escalas de features do dataset e como parametrização inicial, o default foi mantido. A tabela 3 traz os resultados das métricas de avaliações e abaixo temos o gráfico de resíduos com a diferença dos erros.

Modelo	LightGBM	
RMSE	R2 Score	MAE
2.78	81.3%	2.01

Tabela 3 – Resultados LightGBM

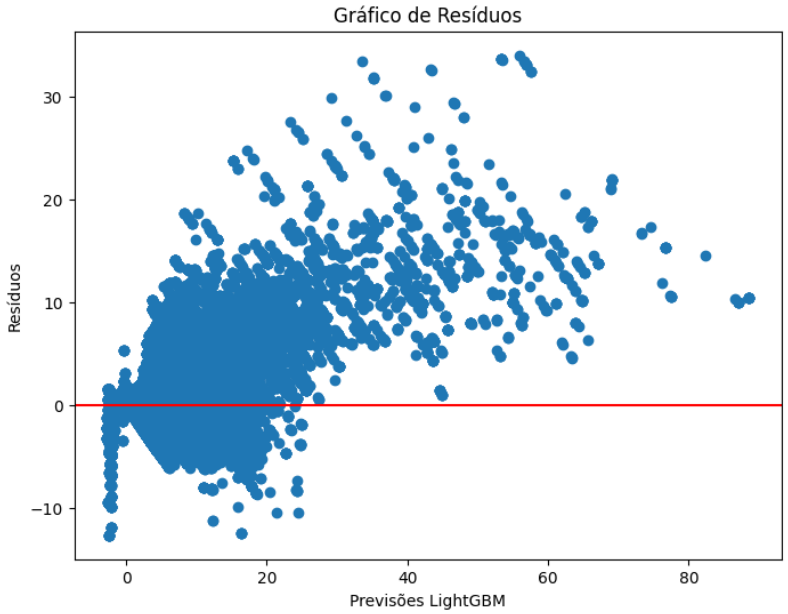

Figura 7 – Gráfico de Resíduos LightGBM

Podemos observar que tivemos um ganho considerável nos resultados das previsões, diminuindo a diferença de erros para aproximadamente 30 jardas apenas, porém, considerando a natureza do problema, este erro ainda não nos permite um bom desempenho na resolução do problema, visto que 30 jardas seriam quase a metade de um campo. Visto da performance dos modelos de gradiente boosting, abaixo teremos uma abordagem semelhante.

3.3.3 XGBoost

Como terceira alternativa de modelo, foi usado o Xgboost que também é um modelo de gradiente boosting, este modelo vem ganhando destaque no mercado devido a sua grande eficiência computacional e capacidade de lidar com conjunto de dados grandes e complexos. A abordagem do modelo é semelhante à do LightGBM, combinando resultados de diferentes árvores de decisão que se ajustam entre iterações para diminuição dos erros e otimização do resultado do modelo final. Para nosso cenário, ajustamos apenas os parâmetros *max_depth* para 20, *eta* para 0.1 e no *eval_metric*, utilizamos o rmse. A tabela 4 mostra os resultados iniciais do modelo e logo abaixo, na figura 7, temos o gráfico de resíduos do modelo.

Modelo	XGBoost	
RMSE	R2 Score	MAE
0.41	99.5%	0.11

Tabela 4 – Resultados XGBoost

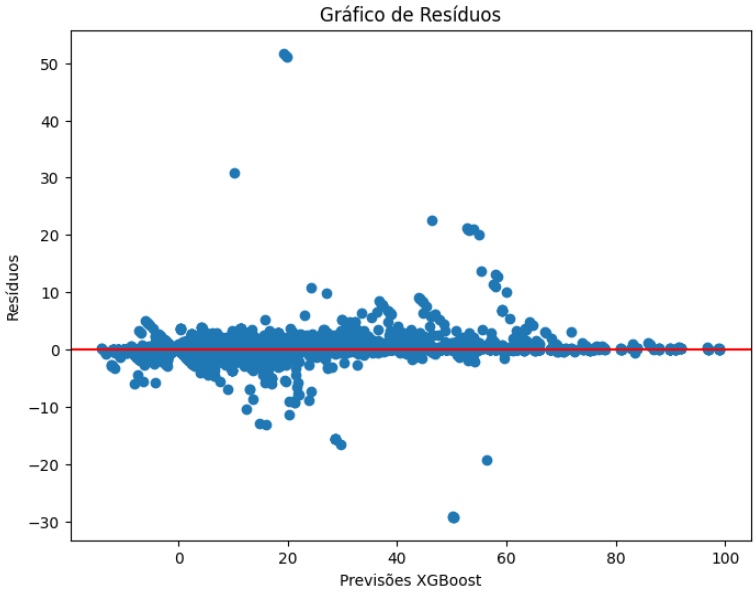

Figura 7 – Gráfico de Resíduos XGBoost

Através dos resultados obtidos e os resíduos dos erros, podemos perceber que este modelo obteve resultados superiores, com R2 Score chegando a 99%, embora ainda exista poucos erros de quase 50 jardas, são casos isolados. Com este resultado houve indagação sobre um possível data leakage, ou seja, uma correlação muito forte entre uma variável específica que, sozinha, acaba por explicar o problema como um todo, por este motivo, houve uma investigação sobre este fator. Como primeiro passo, pegamos a importância de cada feature no modelo original.

Conseguimos identificar uma disparidade de importância muito alta entre as variáveis que descrevem a posição relativa dentro de campo antes da jogada acontecer, seguindo esta lógica, coletamos todas as variáveis que possam descrever o posicionamento do jogador, visto que é possível existir uma forte correlação com a quantidade de jardas que foram ganhas, pois esta seria apenas a diferença entre a posição de início e de final da jogada. Como novo teste, removemos as variáveis X, Y, S, A, Dir, DisplayName, Dis, YardLine, NflIdRusher, GameClock, Orientation, Possession Team, NflId, FieldPosition, JerseyNumber, PlayerCollegeName, PlayerBirthDate, PlayerHeight, PlayerWeight, HomescoreBeforePlay, VisitorScoreBeforePlay.

Na figura 8 podemos analisar a distribuição da importância de features no primeiro treinamento.

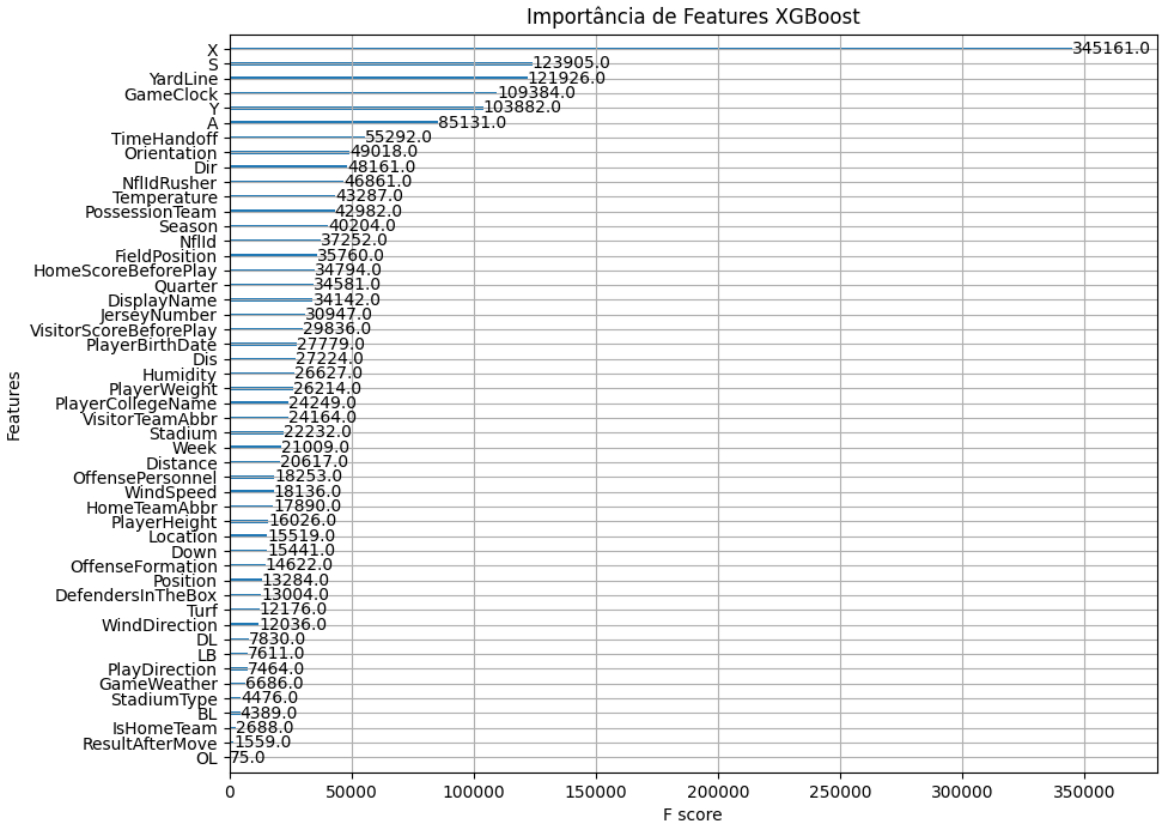

Figura 8 – Importância de Features no modelo.

Após o treinamento de novo modelo, utilizando o dataset sem os dados supracitados, conseguimos manter uma boa taxa de acerto de valores, porém notamos que agora o modelo possui certas confusões, aumentado levemente o número de casos incorretos, como podemos observar no gráfico de resíduos da figura 9.

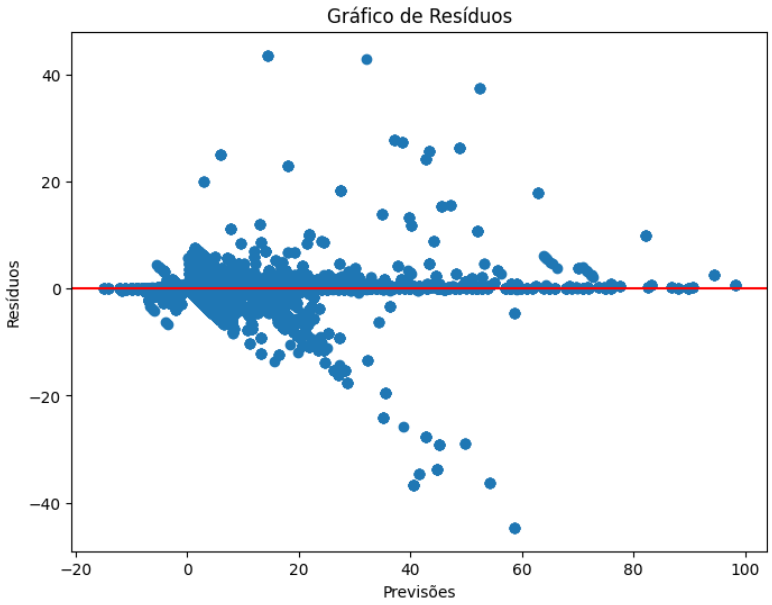

Figura 9 – Gráfico de Resíduos XGBoost pós Feature Selection

Outra diferença notável nos resultados está nas métricas de avaliação, onde a tabela 5 representa os resultados obtidos. Após esta seleção de features, acabamos por generalizar o modelo, onde removemos todas as informações sobre posicionamento relativo de jogadores e informações pessoais de cada atleta, como altura, peso etc., desta forma, baixamos o percentual de acerto, porém ainda conseguimos um modelo que nos de alta probabilidade de acerto sem necessariamente ter os jogadores titulares em campo.

Modelo	XGBoost - Feature Selection	
RMSE	R2 Score	MAE
1.25	96,2%	0.46

Tabela 5 – Resultados XGBoost pós Feature Selection

A figura 10 representa, de uma forma mais comparativa, os resultados previstos com o modelo após aplicação do feature selection versus os resultados reais das jogadas no dataset de teste. Podemos observar que alguns erros tendem para baixo do real e outros acima do real, porém olhando o todo, caracterizam pouquíssimos erros.

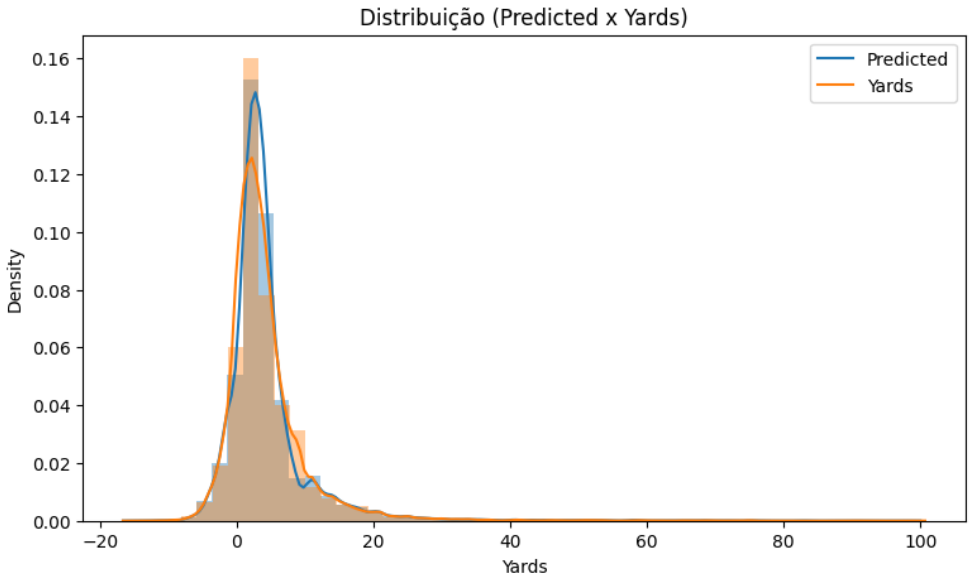

Figura 10 – Comparação entre valores previstos e valores reais.

Outra abordagem utilizada para validação dos resultados após a seleção de features do conjunto, foi a utilização da validação cruzada. Com esta técnica, através do método *model_selection.cross_val_score()* da biblioteca Sklearn, podemos testar o desempenho do modelo com diferentes frações do dataset, garantindo que todas as partes sejam utilizadas como treino e como teste.

Após avaliação com este método, utilizando 5 Folds, conseguimos atestar que os resultados permaneceram com percentual de erros e acertos próximos do validado anteriormente, tendo uma média de 99% de R2 Score. A Figura 11 demonstra os resíduos gerados pelo modelo treinado com validação cruzada.

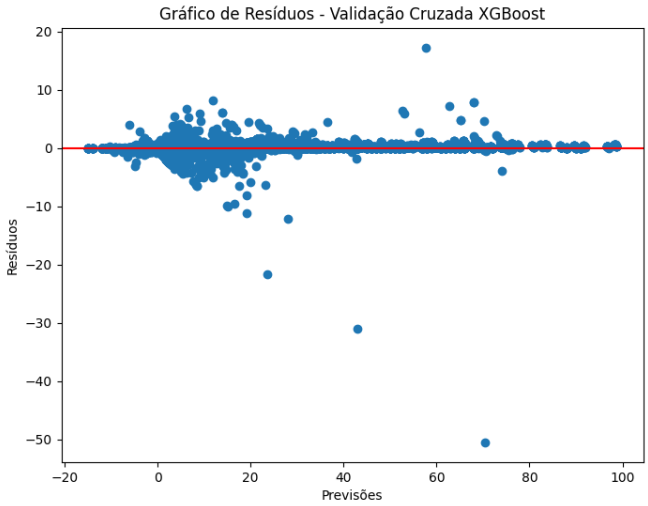

Figura 11 – Gráfico de Resíduos XGboost com validação cruzada

4. Conclusão

Após todas as avaliações realizadas, podemos concluir que o machine learning pode sim ser um grande aliado na construção do playbook e estas técnicas podem trazer bons resultados dentro de uma temporada, como já tem sido explorado por vários times profissionais dentro da liga. O maior desafio ainda se concentra na obtenção de dados mais específicos, na visão de um time, por exemplo, é difícil conseguir dados sobre formações defensivas dos adversários, pois isso é o que mantem a competitividade do jogo.

Ainda analisando resultados do modelo XGBoost, notamos que a formação do time durante o ataque é um fator bem importante para o sucesso, visto que este atributo foi um dos principais, visualizando as importâncias de variáveis no modelo.

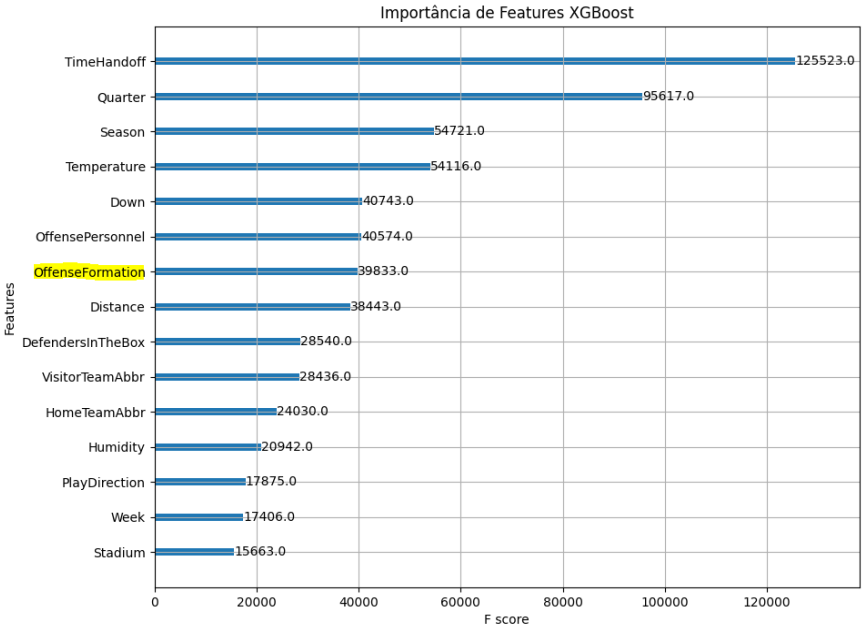

Figura 11 – Top 15 variáveis mais importantes para o modelo.

Pensando em uma evolução do tema, podemos obter melhores resultados combinando dados de estatísticas pessoais de cada jogador, olhando por exemplo para a temporada atual, percentual de recebimento de passes para cada jogador do ataque, percentual de jardas por passe completo, percentual de lançamentos que resultaram em um bom avanço etc. Contudo, uma grande dificuldade deste tema, é a construção do dataset com tais informações, além do fato mencionado anteriormente de alguns dados serem restritos apenas ao time, existe também a periodicidade de atualização destes dados, estatísticas de um jogador podem mudar rapidamente, lesões podem ocorrer e o desempenho de um jogador ou de um time inteiro pode cair, novos jogadores podem integrar um time e acabar mudando a dinâmica do jogo, dentre tantos outros fatos que o machine learning ainda não consegue prever.

Uma outra abordagem que resultaria em bons resultados, é o incremento de uma rede neural para a previsão das jardas por jogadas, se aliado com estas estatísticas pessoais, pode retornar um percentual de acerto muito alto, porém conforme subimos o nível dos resultados, também sobe o preço por isso, o treinamento destas redes neurais precisariam rodar durante várias épocas, pois é o que determinaria um bom resultado. Como exemplo, aplicando o dataset atual deste projeto em uma rede simples, rodando por apenas 10 épocas, levou mais de 3 horas para execução com um resultado bem abaixo dos apresentados neste artigo.

4. Referências

Withoeft, R. J. (1996). Além do Número de Finalizações: Criação e Aplicação de um Modelo de Estimação de Gols Esperados (xG). Universidade Federal do Paraná. Disponível em: https://acervodigital.ufpr.br/handle/1884/71067

Curti, Antony. Manual do Futebol Americano. 2ª edição. Editora Action Books, 2017.

Purucker, M. C. (2020). “Neural network quarterbacking”. University of Pittsburgh. Disponível em: https://ieeexplore.ieee.org/document/535226

Landers, J. R. and Duperrouzel, B. (2019) "Machine Learning Approaches to Competing in Fantasy Leagues for the NFL". Disponível em: https://ieeexplore.ieee.org/document/8367900

Chen, T. and Guestrin C. (2016) “XGBoost: A Scalable Tree Boosting System”. University of Washington. Disponível em: https://arxiv.org/abs/1603.02754

LightGBM. (s.d.). LightGBM Documentation. Recuperado em 20 de Janeiro de 2023, de https://lightgbm.readthedocs.io/en/latest/

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2017). Cap. 3. An Introduction to Statistical Learning with Applications in R. Springer. Disponível em: https://hastie.su.domains/ISLR2/ISLRv2_website.pdf

Pelanek, R. (2015) “Metrics for Evaluation of Student Models” . Masaryk University Disponível em: https://eric.ed.gov/?id=EJ1115277