PREVISÃO DE VENCEDOR DE JOGOS DA SÉRIE A DO CAMPEONATO BRASILEIRO COM RANDOM FOREST

DISCENTES:
FERNANDO LUCAS SOUSA SILVA
TEÓPHILO VITOR DE CARVALHO CLEMENTE

DOCENTE:
ADRIÃO DUARTE DÓRIA NETO

NATAL/RN
2022

RESUMO

O presente trabalho visa apresentar uma solução para um problema de classificação utilizando o algoritmo de aprendizagem de máquina (Machine Learning) - Random Forests - para a previsão de resultados dos jogos da série A do campeonato brasileiro de futebol. Para isso, foi utilizado uma base de dados com jogos e resultados de 2003 a 2022, aplicado um modelo baseado no algoritmo de Inteligência Artificial e coletado as devidas métricas para análise da eficácia do projeto.

Palavras-chaves: Inteligência Artificial, Machine Learning, Random Forests, Futebol, Previsão.

1. INTRODUÇÃO

A inteligência artificial (IA) tem sido bastante utilizada recentemente em vários setores da tecnologia, tornando diversas aplicações mais robustas e inteligentes com diversos algoritmos que podem ser utilizados para diferentes finalidades. Assim, dentre eles, destaca-se o algoritmo de Random Forest que foi registrado por Leo Breiman e Adele Cutler.

Além disso, como citado por Leo Breiman - um dos autores do algoritmo - Florestas Aleatórias (Random Forests) são uma combinação de preditores de árvores, de modo que cada árvore depende dos valores de um vetor aleatório amostrado independentemente e com a mesma distribuição para todas as árvores da floresta.

Com isso, o presente trabalho visa apresentar uma solução para um problema de classificação para a previsão de resultados dos jogos da série A do campeonato brasileiro de futebol utilizando técnicas de IA. Com isso, para esse projeto, utilizou-se o algoritmo de classificação Random Forest que é um algoritmo de aprendizagem de máquina (Machine Learning) amplamente utilizado para problemas de classificação devido a sua facilidade de implementação e flexibilidade (IBM).

Ademais, o problema tem como base de dados resultados dos jogos desde o ano que se iniciou os pontos corridos (2003) até a última temporada até o momento em que esse trabalho foi desenvolvido (2022). Assim, foi possível ter uma grande quantidade de dados para realização dos passos necessários à construção do modelo e consequentemente melhores resultados.

2. FUNDAMENTAÇÃO TEÓRICA

Para o desenvolvimento deste trabalho utilizou-se de pesquisas no material didático disponibilizado pelo professor, nas referências bibliográficas por ele citadas e em artigos que versassem sobre o assunto. Ademais, para embasamento teórico das técnicas utilizadas na construção do projeto, vamos a seguir apresentar brevemente seus fundamentos e como funcionam.

2.1 RANDOM FOREST

O algoritmo de Random Forest, ou floresta de árvores, como o próprio nome diz, consiste num conjunto de várias árvores de decisão que em conjunto permitirão que o algoritmo funcione. Inicialmente precisamos entender como funcionam as árvores de decisão, algoritmo bastante conhecido e com grandes aplicações. As árvores de decisão, estabelecem regras para tomada de uma decisão, sendo assim, ela criará uma estrutura similar a uma árvore, onde os galhos serão nossos "nós", assim ao percorrer tal nó ela fará a verificação de uma condição, e caso seja atendida o fluxo segue por um ramo, caso contrário, por outro ramo, sempre levando ao próximo nó, até a finalização da árvore.

Dado isso, nota-se que o algoritmo funciona tendo a resposta de diversos modelos (árvores distintas) e a partir destas respostas iremos gerar só uma final. Para iniciar o processo os dados a serem usados são escolhidos de forma aleatória e não todos, somente uma parte, neste caso, inicialmente ele irá selecionar randomicamente duas ou mais variáveis e irá a partir de cálculos definir qual será o primeiro nó, para o nó seguinte serão selecionadas mais duas ou mais variáveis, excluindo-se as anteriormente usadas e assim feito os cálculos, esse processo se repetirá até a construção do último nó.

Tendo esse processo definido, agora ocorrerá a replicação finitas vezes deste processo e com isso teremos a construção de novas árvores e com resultados diferentes, já que, o processo é randômico. Assim, quantos mais árvores criarmos, melhor será nosso resultado pois teremos mais amostras para obter uma resposta final coerente e assertiva, claro que, a criação de novas árvores deve ser parado em um dado instante, visto que, essa criação tem um custo computacional, então, ao vermos que o resultado não melhora com a criação de mais árvores sabemos que chegamos no limiar de criação. Por fim, para obtenção do resultado final, a depender da aplicação que se está desenvolvendo, pode ser usado uma média ou o valor que mais vezes apareceu como resposta final do algoritmo.

2.2 BIBLIOTECA PANDAS E SKLEARN

A biblioteca Pandas é uma biblioteca do Python criada por Wes McKinney para análise de dados, ela é construída com base em outras duas bibliotecas mais famosas: a Matplotlib para visualização de dados e NumPy para operações matemáticas, sua vasta sua utilização se dá pelo fato do seu alto desempenho e facilidade.

Ela funciona pegando dados, como um arquivo CSV, TSV ou um banco de dados SQL por exemplo, e a partir deles cria um objeto Python com linhas e colunas chamado DataFrame, que se parece muito com uma tabela e de fácil manuseio. Assim, ela é capaz de modelar, reestruturar, concatenar e realizar outras diversas operações fundamentais para dadas necessidades quando se trabalha com dados, desse modo, ela se tornou indispensável para áreas como data science, deep learning, machine learning e outras que trabalham cotidianamente com dados.

Já a biblioteca sklearn é uma biblioteca também da linguagem Python desenvolvida com propósito específico para aplicação prática em machine learning. Esta biblioteca dispõe de ferramentas simples e eficientes para análise preditiva de dados, como é o caso deste projeto, ela se torna reutilizável em diferentes situações na área e foi construída sobre os pacotes NumPy, SciPy e Matplotlib.

O seu funcionamento consiste em etapas que são o pré-processamento, que consiste na etapa mais trabalhosa no desenvolvimento de um modelo de machine learning. Após isso ela trabalhará com a classificação, desenvolvendo modelos capazes de detectar qual categoria pré-determinada um elemento pertence. Posterior é feita a regressão, que consiste no desenvolvimento de modelos que podem atribuir um valor contínuo a um elemento. Continuando temos a clusterização, onde é desenvolvido um modelo para detecção automática de grupos com características similares em seus integrantes. Feito isso, teremos a redução de dimensionalidade, ela reduzirá o número de variáveis em um problema, com isso, podemos diminuir consideravelmente a quantidade de cálculos necessários em um modelo, aumentando a eficiência, com uma perda mínima de assertividade. Por fim, ela realizará o ajuste de parâmetros, consiste em comparar, validar e escolher parâmetros e modelos, de maneira automatizada, assim é possível facilmente comparar diferentes parâmetros no ajuste de um modelo, encontrando assim a melhor configuração para a aplicação em questão.

Todas as funções acima citadas podem ser implementadas facilmente no nosso problema em questão com a utilização da biblioteca, com isso facilitando de forma notória o desenvolvimento, inclusive no nosso caso usando Random Forest.

3. APLICAÇÃO

A aplicação se dá através da implementação de um algoritmo com auxílio do Colab[1] para execução e compilação, utilizando a linguagem de programação Python e métodos de IA, para a previsão de resultados dos times mandantes em seus jogos do campeonato brasileiro. O link com o algoritmo completo pode ser acessado em: https://colab.research.google.com/drive/1dfnixrrROUlEu6K0TXcpfgNP-9wE9zSc?usp=sharing.

3.1 IMPORTAÇÃO DAS BIBLIOTECAS

Para iniciar o projeto deve-se importar as bibliotecas e ferramentas que facilitarão o desenvolvimento do algoritmo, entretanto, não é uma regra, a medida que for sendo necessário a importação pode-se fazer durante o desenvolvimento em si do código. Assim, foram úteis as bibliotecas: Pandas e Sklearn, como mostrado na figura abaixo:

```
[1]  import pandas as pd
     from sklearn.ensemble import RandomForestClassifier
     from sklearn.metrics import accuracy_score
     from sklearn.metrics import precision_score
```

Figura 1 - Importação das bibliotecas e ferramentas.

3.2 LEITURA DO DATASET

Após a realização das importações, é preciso ingressar a base de dados para o arquivo no Colab onde está sendo desenvolvido o código e, assim, começar a realização do tratamento dos dados e análises. Desse modo, a leitura dos dados é feita de acordo com a figura 2:

```
[2]  dataset = pd.read_csv("campeonato-brasileiro-full.csv", index_col = 0)
     dataset.head()
```

Figura 2 - Leitura dos dados.

Após isso, para melhor entendimento foi analisado o tamanho dos dados de acordo com a quantidade de times e partidas totais que teria que ter no dataset, no entanto, verificando em linhas de código, foi constatado que esse valor não é compatível devido aos clubes que são rebaixados e portanto não participam de todas as edições da Série A do Campeonato

[1] Olá, este é o Colaboratory - Colaboratory (google.com)

Brasileiro, apesar de não conter todos os clubes devido a extensa lista, é possível visualizar a diferença de partidas, como mostrado a seguir:

```
[3]  dataset.shape

     (8025, 15)

[4]  dataset["mandante"].value_counts()

     Sao Paulo          390
     Santos             390
     Fluminense         390
     Flamengo           389
     Internacional      371
     Athletico-PR       371
     Corinthians        371
     Atletico-MG        371
     Gremio             350
     Palmeiras          348
     Cruzeiro           333
     Botafogo-RJ        330
     Vasco              295
     Goias              276
     Coritiba           276
     Figueirense        219
     Sport              209
     Vitoria            198
     Bahia              194
```

Figura 3 - Dados faltantes de clubes rebaixados.

Então, faz-se o tratamento dos dados que serão úteis para o treinamento do modelo utilizando Random Forest, através de modificar o seu “tipo”, alguns dados estavam configurados em formatos não compatíveis para a análise como valores de: data, time visitante e time mandante. Dito isso, foram inseridas 3 colunas novas que se referem a esses dados, mas com o tipo correto.

Outrossim, uma coluna denominada “time” que referencia a do identificador do time visitante e da coluna que mostra o vencedor, que funciona da seguinte maneira: é denominado 0, caso o time mandante perca ou empate e 1, caso o time mandante vença a partida. Por conseguinte, todos esses passos estão exibidos a seguir:

```
[5]  dataset["data"] = pd.to_datetime(dataset["data"])
     dataset["cod_visitante"] = dataset["visitante"].astype("category").cat.codes
     dataset["cod_mandante"] = dataset["mandante"].astype("category").cat.codes
     dataset["time"] = 0
     dataset["hora_int"] = dataset["hora"].str.replace(":.+", "", regex=True).astype("int")
     dataset["cod_dia"] = dataset["data"].dt.dayofweek

[6]  for i in range(1, 8026):
       if dataset.at[i, "mandante"] == dataset.at[i, "vencedor"]:
         dataset.at[i, "vencedor"] = 1
       else:
         dataset.at[i, "vencedor"] = 0

[7]  for i in range(0, 8026):
       for j in range(1, 8026-i):
         if i == dataset.at[j, "cod_visitante"]:
           dataset.at[j, "time"] = i

[8]  dataset["cod_vencedor"] = dataset["vencedor"].astype("category").cat.codes
```

Figura 4 - Tratamento dos dados.

3.3 CONSTRUÇÃO DO MODELO

O modelo baseado em Random Forest é necessário definir a quantidade de estimadores, o número mínimo de amostras e o estado aleatório, além dos dados de treinamento e teste que foram definidos como todos os jogos antes de 2022 como treinamento e os jogos posteriores como teste. Na figura abaixo é mostrado os passos relatados, além das colunas utilizadas como preditores:

```
[10] rf = RandomForestClassifier(n_estimators=100, min_samples_split=2, random_state=1)

[11] train = dataset[dataset["data"]<"2022-01-01"]

[12] train.shape

     (7645, 21)

[13] test = dataset[dataset["data"]>"2022-01-01"]

[14] test.shape

     (380, 21)

[15] predictors = ["cod_mandante", "cod_visitante", "hora_int", "cod_dia"]

[16] rf.fit(train[predictors], train["cod_vencedor"])

     RandomForestClassifier(random_state=1)

[17] preds = rf.predict(test[predictors])
```

Figura 5 - Modelo Random Forest.

Assim, é possível visualizar a acurácia e precisão do modelo de classificação construído, que resultou em 51,31% e 45,45%, o que não é tão satisfatório, e por isso, foi desenvolvida duas funções para aprimoramento desses resultados, retreinando o modelo com duas novas colunas que ainda não haviam sido treinadas previamente.

```
[25] def rolling_averages(grupo, cols, new_cols):
        grupo = grupo.sort_values("data")
        rolling_stats = grupo[cols].rolling(3, closed="left").mean()
        grupo[new_cols] = rolling_stats
        grupo = grupo.dropna(subset=new_cols)
        return grupo

[26] cols = ["mandante_Placar", "visitante_Placar"]
     new_cols = [f"{c}_rolling" for c in cols]
```

Figura 6 - Função que agrupa pela data e utiliza duas novas colunas.

```
[32] def predicoes(data, predictors):
        train = data[data["data"]<"2022-01-01"]
        test = data[data["data"]>"2022-01-01"]
        rf.fit(train[predictors], train["cod_vencedor"])
        preds = rf.predict(test[predictors])
        combined = pd.DataFrame(dict(atual = test["cod_vencedor"], predicted=preds), index=test.index)
        precision = precision_score(test["cod_vencedor"], preds)
        return combined, precision
```

Figura 7 - Função de predição que retreina o modelo.

Com a utilização dessas duas novas funções o modelo retornou uma leve melhora na métrica de precisão, indo de 45,45% para 53% de média. No entanto, ao aplicar para os clubes que mais participaram da Série A, ou seja, aqueles que não foram rebaixados no período de 2003 a 2022, é possível alcançar uma precisão de aproximadamente 70%. Como mostrado na figura abaixo, o exemplo do Clube de Regatas do Flamengo, nela a coluna “atual” é o resultado real e “predicted” o resultado previsto pelo modelo, em que “0” é derrota ou empate e “1”, vitória:

	atual	predicted	data	mandante	visitante
3655	1	1	2022-01-10	Flamengo	Bragantino
3656	0	0	2022-02-11	Flamengo	Corinthians
3657	0	1	2022-04-09	Flamengo	Ceara
3658	1	1	2022-04-17	Flamengo	Sao Paulo
3659	0	1	2022-04-20	Flamengo	Palmeiras
3660	0	1	2022-05-06	Flamengo	Fortaleza
3661	0	0	2022-05-10	Flamengo	Internacional
3662	1	0	2022-05-21	Flamengo	Goias
3663	1	1	2022-06-15	Flamengo	Cuiaba
3664	1	1	2022-06-25	Flamengo	America-MG
3665	1	1	2022-07-16	Flamengo	Coritiba
3666	1	1	2022-07-20	Flamengo	Juventude
3667	1	1	2022-07-30	Flamengo	Atletico-GO
3668	0	1	2022-08-05	Flamengo	Botafogo-RJ
3669	1	1	2022-08-14	Flamengo	Athletico-PR
3670	0	1	2022-09-18	Flamengo	Fluminense
3671	1	1	2022-10-15	Flamengo	Atletico-MG
3672	1	1	2022-10-25	Flamengo	Santos
3673	0	0	2022-12-11	Flamengo	Avai

Figura 8 - Resultado final para um clube específico.

4. CONCLUSÃO

A finalização desse projeto possibilitou obter uma experiência de como desenvolver uma aplicação interessante e prática, que pode ser utilizada para diversos fins, com conceitos e fundamentos da Inteligência Artificial através do algoritmo de Random Forest. Portanto, dando um embasamento teórico e prático mais efetivo a esse método tão importante no ramo da tecnologia, da inteligência artificial e da computação.

5. BIBLIOGRAFIA

[1] Material didático do Professor Adrião.

[2] Breiman, L. **Random Forests**. *Machine Learning* 45, 5–32 (2001).

[3] IBM Cloud Education. **Random Forest**. Acesso a partir de: What is Random Forest? | IBM.

[4] A biblioteca scikit-learn. **Biblioteca SKLEARN Aplicações**. Website Didática.

[5] Biblioteca Pandas. **O que é a biblioteca Pandas?**. Website Voitto.