

**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**  
**NÚCLEO DE EDUCAÇÃO A DISTÂNCIA**  
**Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data**

**PABLO SILVA FONSECA**

**ANÁLISE EXPLORATÓRIA DE DADOS DE FUTEBOL E UTILIZAÇÃO DE  
MODELOS DE MACHINE LEARNING PARA PREVISÃO DE RESULTADOS**

Belo Horizonte

2021

**PABLO SILVA FONSECA**

**ANÁLISE EXPLORATÓRIA DE DADOS DE FUTEBOL E UTILIZAÇÃO DE  
MODELOS DE MACHINE LEARNING PARA PREVISÃO DE RESULTADOS**

Trabalho de Conclusão de Curso apresentado  
ao Curso de Especialização em Ciência de  
Dados e Big Data como requisito parcial à  
obtenção do título de especialista.

Belo Horizonte

2021

## SUMÁRIO

<b>1. Introdução.....</b>	<b>4</b>
<b>1.1. Contextualização .....</b>	<b>4</b>
<b>1.2. O problema proposto .....</b>	<b>4</b>
<b>2. Coleta de Dados .....</b>	<b>6</b>
<b>3. Processamento/Tratamento de Dados .....</b>	<b>10</b>
<b>4. Análise e Exploração dos Dados .....</b>	<b>19</b>
<b>5. Criação de Modelos de Machine Learning .....</b>	<b>19</b>
<b>6. Apresentação dos Resultados .....</b>	<b>21</b>
<b>7. Links .....</b>	<b>27</b>

## 1. Introdução

### 1.1. Contextualização

A grande geração e demanda por dados nos dias de hoje também está presente no mundo dos esportes.

No futebol, além das duas variáveis principais (quantos gols cada equipe marcou), outras dezenas, talvez centenas circundam o ambiente. Passes certos, passes errados, chutes a gol, chutes pra fora, faltas cometidas, faltas sofridas, estão de alguma forma relacionadas com o placar final.

O investimento por parte dos clubes em modelos de machine learning se mostra interessante, visto as altas cifras que circundam o ambiente do futebol. O futebol inglês no ano de 2018 teve uma receita de aproximadamente € 5,7 bilhões, onde o time campeão recebeu aproximadamente € 87 milhões.

Existe uma grande e crescente comunidade em torno de estatísticas e previsões de resultados esportivos. Talvez a mais famosa seja *March Madness* no Kaggle, onde especialistas e entusiastas constroem modelos para prever resultados da liga americana de basquete universitário. Em alguns anos a premiação ultrapassa 20 mil dólares.

Estatística dentro do esporte não fica restrito somente ao campo de data Science, em 2011 o filme MoneyBall, traduzido no Brasil como “O Homem que mudou o jogo” mostra *Billy Beane, gerente geral do Oakland A's, um dia tem uma epifania: a sabedoria convencional do beisebol está totalmente errada. Com o uso de estatística, os protagonistas contratam e demitem jogadores a fim de montar um time vencedor. o filme recebeu várias indicações ao Oscar, incluindo de melhor filme.*”

### 1.2. O problema proposto

Os campeonatos nacionais de futebol possuem algumas vantagens quando comparados à eventos esporádicos, como copa do mundo e olímpiadas. Alguns dados não variam tanto de um jogo para outro, temos sempre um time visitante contra o time da casa. Ocorrendo esta situação dez vezes por rodada, resultando

em um total de 380 partidas por ano. Considerando a semelhança no formato do campeonato, temos dezenas de países gerando dados. Com certa facilidade conseguimos atingir um dataset de milhares de linhas.

Nas ligas disputadas dentro de um único país, como a NBA (basquete) nos Estados Unidos, Campeonato Brasileiro de Futebol (Brasileirão) no Brasil, e as ligas Europeias (objeto de estudo deste trabalho) é marcado por um confronto entre o time da casa contra o time visitante. Se tratando de um jogo igualitário, sem favorecimento para o time da casa, é esperado que no longo prazo encontremos algo como um terço de vitórias para o time da casa, um terço de empates e um terço de derrotas.

Mas uma pessoa que acompanha minimamente esportes tem a percepção que o time da casa tem mais resultados positivos que negativos, talvez por jogar com o apoio de sua torcida, em um campo em que realiza os treinamentos, não tem o cansaço da viagem, está acostumado com as condições climáticas da cidade, além de tantas outras variáveis não tão óbvias. E essa percepção pode ser facilmente confirmada, em uma análise simplesmente quantitativa, podemos observar que no agregado do Campeonato Inglês de 2005 de 2018 o time da casa venceu 47 % dos jogos, outros 25% terminaram empatados, e somente 29% foram derrotados.

Neste trabalho serão utilizados dados de duas ligas nacionais, English Premier League (Campeonato Inglês) e Serie A (Campeonato Italiano). Os dados são referentes do período 2005 à 2018.

O objetivo deste trabalho é criar um modelo que seja capaz de fazer previsões sobre o vencedor de um confronto entre duas equipes de futebol. Utilizando dados do desempenho das equipes nos jogos anteriores.

## 2. Coleta de Dados

### 2.1 Dados Football-Data

Os dados foram coletados do site Football-Data <http://football-data.uk.co>, um site especializado em estatísticas esportivas, com o objetivo no auxílio de tomadas de decisão para realizar apostas esportivas. Este trabalho não irá abordar a temática apostas, mas os dados disponibilizados também são úteis para outros campos de estudo.

Os dados são separados por países e temporadas. Onde cada temporada é um arquivo do tipo CSV.

Os campeonatos europeus de futebol em sua maioria possuem 20 clubes participantes, isso totaliza um total de 380 confrontos, que é o número de linhas de cada um dos datasets. Foram importados 28 datasets do Football-Data, totalizando um total de 10640 linhas.

Cada um dos 28 datasets possuem 60 colunas, que estão descritas no Quadro 1 abaixo.

Nome da coluna/campo	Descrição	Tipo
Div	Campeonato	object
Date	Data de realização da partida	Object
HomeTeam	Time que está jogando em casa	Object
AwayTeam	Time Visitante	Object
FTHG	Qtde gols marcados pelo time da casa	Int64
FTAG	Qtde gols marcados pelo visitante	Int64
FTR	Resultado Final	Int64
HTHG	Qtde de gols marcados pelo time da casa até o intervalo	Int64

HTAG	Qtde de gols marcados pelo visitante até o intervalo	Int64
HTR	Resultado no intervalo	Int64
Referee	Árbitro da partida	str
HS	Chutes do time casa	Int64
AS	Chutes do time Visitante	Int64
HST	Chutes no alvo	Int64
AST	Chuts no alvo visitante	Int64
HF	Faltas time da casa	Int64
AF	Faltas time visitante	Int64
HC	Escanteios time casa	Int64
AC	Escanteios time visitante	Int64
HY	C. Amarelos time casa	Int64
AY	C. Amarelos time visitante	Int64
HR	C. Vermelhos time casa	Int64
AR	C. Vermelhos time visitante	Int64
B365 e outros	Odds para apostas esportivas	Int64

Quadro 1 – Colunas de cada dataset

Check out an [example datasheet](#) and a set of [Notes](#) that describe the available data. The table below provides quick links to all the data files, with descriptions of exactly what data can be found in each data file. Data files can also be accessed via the [country links](#) in the right hand menu.

#### Main Leagues












 <a href="#">England Football Results</a>	Premiership & Divs 1,2,3 & Conference
 <a href="#">Scotland Football Results</a>	Premiership & Divs 1,2 & 3
 <a href="#">Germany Football Results</a>	Bundesligas 1 & 2
 <a href="#">Italy Football Results</a>	Serie A & B
 <a href="#">Spain Football Results</a>	La Liga (Primera & Segunda)
 <a href="#">France Football Results</a>	Le Championnat & Division 2
 <a href="#">Netherlands Football Results</a>	KPN Eredivisie
 <a href="#">Belgium Football Results</a>	Jupiler League
 <a href="#">Portugal Football Results</a>	Liga I
 <a href="#">Turkey Football Results</a>	Ligi 1
 <a href="#">Greece Football Results</a>	Ethniki Katigoria

Figura 2.1 – Datasets disponíveis no site Football-Data

## 2.2 Dados da classificação do ano anterior

Com o objetivo de enriquecer o conjunto de dados, foram inseridos dados de um segundo dataset. Este dataset possui informações sobre a classificação do time na temporada anterior. Com estes dados podemos trabalhar com a hipótese de que um clube que fez uma boa temporada passada tende a manter um bom desempenho.

Os dados foram inseridos manualmente no Excel após consulta aos sites oficiais da Premier League e Serie A. Posteriormente foi gerado um arquivo CSV para ser utilizado no modelo.

Na Figura 2.2, linha 0, podemos verificar que o Arsenal ficou na posição 2 no ano 2005, 4 nos anos 2006 e 2007, e assim sucessivamente. Os valores “NaN” indicam que o clube não disputou a posição naquele ano.

```
[48] #Importação de um segundo dataset para complementar
```

```
classificacao = pd.read_csv('https://raw.githubusercontent.com/pablofonseca14/projetotcc/main/dataset/claf_anos_a')
classificacao
```

	TEAM	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018
0	Arsenal	2.0	4.0	4.0	3.0	4.0	3.0	4.0	3.0	4.0	4.0	3.0	2.0	5.0	6.0
1	Ascoli	NaN	10.0	19.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	Aston Villa	10.0	16.0	11.0	6.0	6.0	6.0	9.0	16.0	15.0	15.0	17.0	20.0	NaN	NaN
3	Atalanta	20.0	NaN	8.0	9.0	11.0	18.0	NaN	12.0	15.0	11.0	17.0	13.0	4.0	7.0

Figura 2.2 – Fração dataset com a classificação de cada clube no respectivo ano

Os clubes deixam de disputar o campeonato quando no ano anterior terminam nas 3 últimas posição.

Então fiz a opção de substituir os valores “NaN” pelo número “17”, que seria a última posição possível para este clube no ano anterior.

A Figura 2.3 mostra a substituição dos valores “NaN” utilizando *fillna()*, e a inserção das colunas “Casa\_AA”, posição do time da casa no Ano Anterior, e a coluna “Visitante\_AA”, posição do time Visitante no Ano Anterior.



```
classificacao.set_index(['TEAM'], inplace=True)
classificacao = classificacao.fillna(17)

def ano_anterior(campeonato, classificacao, ano):
    casa_AA = []
    visitante_AA = []
    for i in range(380):
        ht = campeonato.iloc[i].Casa
        at = campeonato.iloc[i].Visitante
        casa_AA.append(classificacao.loc[ht][ano])
        visitante_AA.append(classificacao.loc[at][ano])
    campeonato['Casa_AA'] = casa_AA
    campeonato['Visitante_AA'] = visitante_AA
    return campeonato
```

Figura 2.3 – Classificação do Ano Anterior

### 3. Processamento/Tratamento de Dados

Para o processamento, tratamento e análise de dados será utilizado linguagem Python, via Google Colab.

Este trabalho se baseia em analisar o retrospecto dos clubes antes do confronto, para isso as colunas a serem utilizadas neste momento são: 'Div', 'Date', 'HomeTeam', 'AwayTeam', 'FTHG', 'FTAG', 'FTR'.

A figura 3.1 mostra a atualização dos datasets, deixando somente as colunas de interesse. Agora é esperado que cada dataset possua 380 linhas e 7 colunas.

```
principais_colunas = ['Div', 'Date', 'HomeTeam', 'AwayTeam', 'FTHG', 'FTAG', 'FTR']

eng_18_19 = eng_18_19[principais_colunas]
eng_17_18 = eng_17_18[principais_colunas]
eng_16_17 = eng_16_17[principais_colunas]
eng_15_16 = eng_15_16[principais_colunas]
eng_14_15 = eng_14_15[principais_colunas]
eng_13_14 = eng_13_14[principais_colunas]
eng_12_13 = eng_12_13[principais_colunas]
eng_11_12 = eng_11_12[principais_colunas]
eng_10_11 = eng_10_11[principais_colunas]
eng_09_10 = eng_09_10[principais_colunas]
eng_08_09 = eng_08_09[principais_colunas]
eng_07_08 = eng_07_08[principais_colunas]
eng_06_07 = eng_06_07[principais_colunas]
eng_05_06 = eng_05_06[principais_colunas]
ita_18_19 = ita_18_19[principais_colunas]
ita_17_18 = ita_17_18[principais_colunas]
ita_16_17 = ita_16_17[principais_colunas]
ita_15_16 = ita_15_16[principais_colunas]
ita_14_15 = ita_14_15[principais_colunas]
ita_13_14 = ita_13_14[principais_colunas]
ita_12_13 = ita_12_13[principais_colunas]
ita_11_12 = ita_11_12[principais_colunas]
ita_10_11 = ita_10_11[principais_colunas]
ita_09_10 = ita_09_10[principais_colunas]
ita_08_09 = ita_08_09[principais_colunas]
ita_07_08 = ita_07_08[principais_colunas]
ita_06_07 = ita_06_07[principais_colunas]
ita_05_06 = ita_05_06[principais_colunas]
```

Figura 3.1 – Escolha das colunas de interesse

Para facilitar o entendimento foi traduzido o nome das colunas a serem utilizadas. Onde agora temos:

- HomeTeam > Casa
- AwayTeam > Visitante
- FTHG > Placar Casa
- FTAG > Placar Visitante
- FTR > Resultado

### 3.1 Limpeza de dados

Após a escolha das colunas de interesse, foi feita uma análise nas linhas dos datasets. Primeiramente utilizando a função `.shape()`, foi verificado que 4 dos 28 datasets possuíam mais de 380 linhas. Essa informação se mostrou incoerente, como todos os campeonatos tem o mesmo formato estes valores não são possíveis.

Continuando a análise da qualidade dos dados, foi verificado a presença de valores “NaN” em 4 dos 28 datasets, sendo estes os mesmos que tiveram problemas quando consultados com a função `.shape()`.

Foi feito então a verificação destas linhas, e posteriormente a remoção das mesmas. Conforme Figura 3.2

<code>for i in datasets: print(i.shape)</code>	<code>for i in datasets: print(i.isnull().values.any())</code>
(380, 7)	False
(380, 7)	False
(380, 7)	False
(380, 7)	False
(381, 7)	True
(380, 7)	False
(380, 7)	False
(380, 7)	False
(380, 7)	False
(380, 7)	False
(380, 7)	False
(380, 7)	False
(380, 7)	False
(380, 7)	False
(380, 7)	False
(380, 7)	False
(380, 7)	False
(381, 7)	True
(381, 7)	True
(380, 7)	False
(383, 7)	True
(380, 7)	False
(380, 7)	False
(380, 7)	False
(380, 7)	False
(380, 7)	False
(380, 7)	False
(380, 7)	False
(380, 7)	False

Figura 3.2 – Identificação de linhas “NaN”

```
eng_14_15 = eng_14_15.drop([380])
ita_15_16 = ita_15_16.drop([380])
ita_14_15 = ita_14_15.drop([380])
ita_12_13 = ita_12_13.drop([380, 381, 382])
```

Figura 3.3 – Remoção de linhas “NaN”

Um outro erro muito comum e que poderia vir acontecer nestes datasets, seria a presença de linhas duplicadas. Mas foi verificado e nenhum dos datasets possuíam esse tipo de inconformidade.

## 3.2 Tratamento de Dados

### 3.2.1 Comportamento das equipes jogando em casa e como visitante

Neste momento cada linha de cada um dos 28 datasets, só possuem informações do confronto atual. Nosso modelo visa realizar uma previsão de resultados baseado no histórico dos dois times antes do confronto.

Então, utilizando linguagem python, predominantemente as bibliotecas pandas e numpy, foram construídas novas colunas que contenham dados acumulados das rodadas anteriores.

As primeiras colunas a serem inseridas serão de quantos pontos cada uma das equipes marcou neste confronto (Casa Vs Visitante), dessa forma poderemos utilizar estes dados para saber qual equipe mais eficiente antes do confronto.

```
def pontos(campeonato):
    condicoes = (campeonato['Placar Visitante'] > campeonato['Placar Casa'], campeonato['Placar Visitante'] < campeonato['Placar Casa'], campeonato['Placar Visitante'] == campeonato['Placar Casa'])
    valores = [0, 3, 1]
    valores2 = [3, 0, 1]
    campeonato['Pontos Casa'] = np.select(condicoes, valores)
    campeonato['Pontos Visitante'] = np.select(condicoes, valores2)

pontos(eng_18_19)
pontos(eng_17_18)
```

Figura 3.4 – Criação das Colunas Pontos Casa e Pontos Visitante

```
def soma_de_pontos(campeonato):
    campeonato['Soma Pontos Casa'] = campeonato.groupby(['Casa'])['Pontos Casa'].cumsum() - campeonato['Pontos Casa']
    campeonato['Soma Pontos Visitante'] = campeonato.groupby(['Visitante'])['Pontos Visitante'].cumsum() - campeonato['Pontos Visitante']

soma_de_pontos(eng_17_18)
soma_de_pontos(eng_18_19)
```

Figura 3.5 – Criação das Colunas Soma Pontos Casa e Soma Visitantes

Com a criação destas novas colunas, já temos o primeiro dado para nossa exploração e aplicação de machine learning. A figura 3.6, mostra que o Stoke marcou 17 pontos quando jogando em casa, contra 12 do Crystal Palace.

```

17.sample(1, random_state=1)
e da casa (Stoke) marcou 1 gol, enquanto o time visitante (Crystal P) marcou 0. Tendo como resultado "C" (Casa)
e Everton ambos marcaram 2 gols, tendo como resultado "E" (Empate)
e City marcou 3 pontos e o Crystal Palace marcou 0 pontos.
deste confronto, jogando em casa, o Stoke City havia marcado 17 pontos. Crystal Palace marcou 12 jogando como visitante.

```

iv	Data	Casa	Visitante	Placar Casa	Placar Visitante	Resultado	Pontos Casa	Pontos Visitante	Soma Pontos Casa	Soma Pontos Visitante
EO	11/02/17	Stoke	Crystal Palace	1	0	C	3	0	17	12

Figura 3.6 – Aplicação das novas colunas

Intuitivamente podemos dizer que o time com mais pontos tende a vencer a partida. Mas várias outras variáveis serão adicionadas para ajudar o modelo a tomar a sua decisão.

O próximo dado a ser inserido é a quantidade de gols que cada equipe marcou e sofreu antes do confronto, primeiramente separando quando jogando em casa e quando jogando como visitante. Para isso foi utilizado `.goupby()`, para agrupar os times e valores desejados e a função `.cumsum()` para obter o valor acumulado antes do confronto.

A figura 3.7, mostra a função e a aplicação em alguns dos datasets;

```

def soma_de_gols(campeonato):
    campeonato['Gols Casa'] = campeonato.groupby(['Casa'])['Placar Casa'].cumsum() - campeonato['Placar Casa']
    campeonato['Gols Visitantes'] = campeonato.groupby(['Visitante'])['Placar Visitante'].cumsum() - campeonato['Placar Visitante']
    campeonato['Gols Conc. Casa'] = campeonato.groupby(['Casa'])['Placar Visitante'].cumsum() - campeonato['Placar Visitante']
    campeonato['Gols Conc. Visitante'] = campeonato.groupby(['Visitante'])['Placar Casa'].cumsum() - campeonato['Placar Casa']

soma_de_gols(eng_18_19)
soma_de_gols(eng_17_18)
soma_de_gols(eng_16_17)

```

Figura 3.7– Criação de colunas relativas ao número de gols

Após a aplicação da função em todos os datasets, foram acrescidas 4 novas colunas;

- “Gols Casa”: Quantos gols a equipe Casa marcou antes do confronto, quando jogando em casa
- “Gols Visitante”: Quantos gols a equipe visitante marcou antes do confronto, quando jogando como visitante.

- “Gols Conc. Casa”: Quantos gols a equipe casa concedeu antes do confronto, quando jogando como visitante.
- “Gols conc. Visitante”: Quantos gols a equipe visitante concedeu antes do confronto, quando jogando como visitante.

Gols Casa	Gols Visitantes	Gols Conc. Casa	Gols Conc. Visitante
15	19	15	25

Figura 3.8. – Novas colunas relativas aos gols de cada equipe

Utilizando o mesmo confronto do exemplo anterior, temos que o Stoke marcou 15 gols e sofreu 15 quando jogando em casa, enquanto o visitante marcou 19 e sofreu 25 gols.

Neste momento temos o histórico de gols e pontos de cada equipe, quando jogando em casa ou no estádio do adversário.

### 3.2.2 Comportamento das equipes ao longo de todo campeonato

Na seção anterior foram inseridas as colunas; “Soma Pontos Casa”, “Soma Pontos Visitante”, “Gols Casa”, “Gols Visitantes”, “Gols Conc. Casa”, “Gols Conc. Visitante”. Estas colunas são relativas ao desempenho do time jogando como mandante e como visitante. Nesta seção serão inseridas colunas relacionadas com o desempenho do time ao longo de todo campeonato.

O próximo dado a ser inserido é a quantidade de gols que cada time marcou até o momento antes do confronto a ser analisado. Com isso foram inseridas 4 novas colunas;

- GMC > Gols Marcados Casa
- GMV > Gols Marcados Visitante
- GCC > Gols Concedidos Casa
- GCV > Gols Concedidos Visitante

A figura 3.9 mostra a criação da função “gols\_marcados”.

```
def gols_marcados(campeonato):
    # Criação de dicionário. key = times
    times = {}
    for i in campeonato.groupby('Casa').mean().T.columns:
        times[i] = []

    # Valor correspondente para cada chave. Cada time terá um GMC ou GMV
    for i in range(380):
        GMC = campeonato.iloc[i]['Placar Casa'] #Gols Marcados Casa
        GMV = campeonato.iloc[i]['Placar Visitante'] #Gols Marcados Visitante
        times[campeonato.iloc[i].Casa].append(GMC)
        times[campeonato.iloc[i].Visitante].append(GMV)

    # Criação de DF; linhas = times, colunas = rodada
    GolsMarcados = pd.DataFrame(data=times, index = [i for i in range(1,39)]).T
    GolsMarcados[0] = 0
    # Somar quantidade de gols da rodada atual com o somatório anterior
    for i in range(2,39):
        GolsMarcados[i] = GolsMarcados[i] + GolsMarcados[i-1]
    return GolsMarcados
```

Figura 3.9 – Função Gols Marcados por Cada Equipe

A função para os dados de gols concedidos por cada equipe é semelhante aos gols marcados.

O próximo passo foi a criação de duas novas colunas;

- Total P. Casa > número de pontos marcados pelo time Casa
- Total P. Visitante > número de pontos marcados pelo time Visitante

Este dado indicará qual time tem o melhor desempenho no campeonato antes do confronto, e pode ser um forte indicador do time favorito a vitória.

As funções para obtenção destes dados têm lógica similar a obtenção dos gols nos exemplos anteriores.

O próximo dado a ser adicionado aos datasets foi a rodada em que este confronto acontece, o objetivo deste dado é utilizá-lo para racionar os atributos de cada time por rodada. Naturalmente nas últimas rodadas do campeonato, os times

possuirão elevados número de gols e pontos quando comparados com o início do campeonato, e essa distorção pode diminuir a taxa de acerto de alguns modelos de Machine Learning.

A lógica empregada foi que a cada 10 jogos temos uma rodada, totalizando 38 rodadas ao longo das 380 linhas do dataset. A lógica empregada para obtenção desta coluna está demonstrada na figura 3.10.

```
def rodada(campeonato):  
    n = 1  
    rodada = []  
    for i in range(380):  
        rodada.append(n)  
        if((i+1)%10) == 0:  
            n = n+1  
    campeonato['Rodada'] = rodada  
    return campeonato
```

Figura 3.10 – Definição da rodada

Outro importante dado foram os resultados passados das duas equipes envolvidas no confronto. Foi utilizada uma função para obter os 3 últimos resultados de cada equipe, onde foram criadas seis novas colunas, três pra cada time.

- C1 – último resultado time casa
- C2 – penúltimo resultado time casa
- C3 – antepenúltimo resultado time casa
- V1 – último resultado time visitante
- V2 – penúltimo resultado time visitante
- V3 – antepenúltimo resultado time visitante

Para representar os resultados foram utilizadas as letras, “S”: Sucesso, “E”: Empate, “D”: Derrota.



```

def res_anteriores(campeonato,num):
    anterior = res_1(campeonato)
    res_anterior = anterior.copy()
    for i in range(num,39):
        res_anterior[i] = ''
        j = 0
        while j < num:
            res_anterior[i] += anterior[i-j]
            j += 1
    return res_anterior

def ad_res(campeonato,num):
    anterior = res_anteriores(campeonato,num)
    h = ['-'] * (num * 10) # Resultado nulo,
    a = ['-'] * (num * 10)

    j = num
    for i in range((num*10),380):
        casa = campeonato.iloc[i].Casa
        visit = campeonato.iloc[i].Visitante

        ultimo = anterior.loc[casa][j]
        h.append(ultimo[num-1])

        ultimo = anterior.loc[visit][j]
        a.append(ultimo[num-1])

        if ((i + 1)% 10) == 0:
            j = j + 1

    campeonato['C' + str(num)] = h
    campeonato['V' + str(num)] = a

    return campeonato

```

Figura 3.11 – Função para resultados anteriores

A figura 3.12 mostra as novas colunas adicionadas. Antes da rodada 25 o time da casa teve um retrospecto de; Derrota-Emapte-Derrota. Enquanto o time visitante teve Derrota-Sucesso-Derrota.

Rodada	C1	V1	C2	V2	C3	V3
25	D	D	E	S	E	D

Também foi criada uma coluna do tipo *string* para concatenar o desempenho de cada um dos times. As colunas “Casa Sequencia” e “Visitante Sequencia” concatenam os desempenhos de cada uma das equipes.

Casa Sequencia	Visitante Sequencia
DEE	DSD

Figura 3.13 – Colunas com o histórico das duas equipes

Com o objetivo de mais dados para uma maior assertividade dos modelos de Machine Learning foram calculadas as diferenças de gols e pontos entre as equipes do confronto.

```
#Diferença de gols
jogos['Dif Gols Casa'] = jogos['Gols Casa'] - jogos['Gols Conc. Casa']
jogos['Dif Gols Visit'] = jogos['Gols Visitantes'] - jogos['Gols Conc. Visitante']
jogos['Dif Gols Time Casas'] = jogos['GMC'] - jogos['GCC']
jogos['Dif Gols Time Visitante'] = jogos['GMV'] - jogos['GCV']

# Diferença de Pontos
jogos['Dif Pontos'] = jogos['Soma Pontos Casa'] - jogos['Soma Pontos Visitante']

# Diferença da posição no ano anterior
jogos['Dif Posição'] = jogos['Casa_AA'] - jogos['Visitante_AA']
```

Figura 3.14 – Novas Colunas representando a diferença entre os dois times

Até este momento cada função foi aplicada individualmente em cada dataset. Então foi feita a concatenação dos dados através da biblioteca Pandas para os dados serem trabalhados posteriormente na sessão de Machine Learning e Análise Exploratória dos Dados.

#### 4. Análise e Exploração dos Dados

A exploração de dados deste trabalho focou em observar a quantidade de partidas vencidas pela equipe da Casa e pela equipe visitante.

O primeiro estudo feito realizando uma observação em todo o dataset construído no Capítulo 3 deste trabalho, revelou que o time que está jogando em casa, vence 46% das partidas, contra 26% de empates, e 28% derrotas.

```
colors = ["#e74c3c", "#34495e", "#2ecc71"]
colors1 = sns.color_palette("hls", 3)
plt.figure(figsize=(10,10))
df["Resultado"].value_counts().plot.pie(autopct = "%1.0f%",
                                         colors = colors,
                                         wedgeprops = {"linewidth":2,"edgecolor":"white"},
                                         textprops = {"fontsize": 15})
plt.title("Resultados", fontsize = 15)
plt.ylabel("");
```

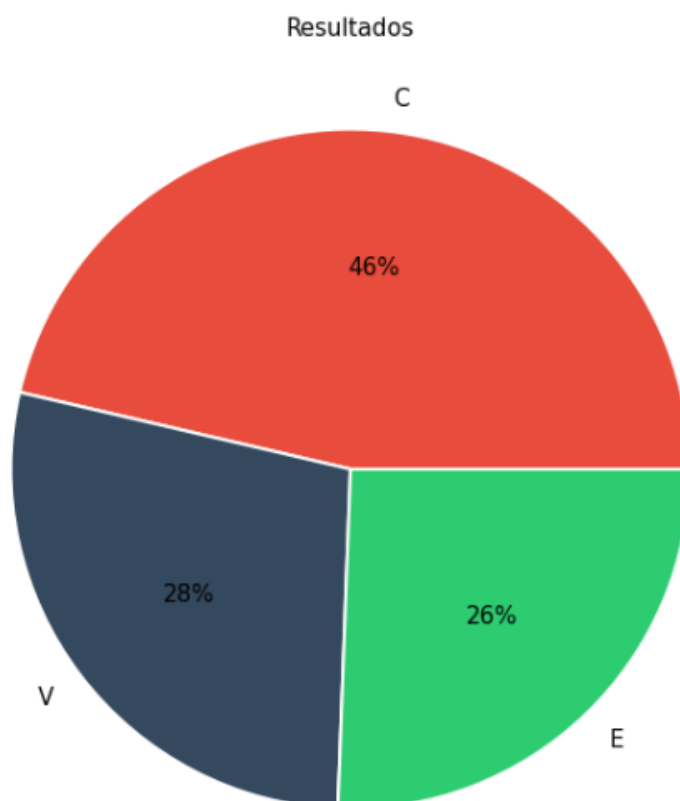


Figura 4.1 – Distribuição de Resultados em todo Dataset

Neste trabalho estão sendo utilizados dados de dois países, Inglaterra e Itália. Então, se mostrou interessante observar o comportamento da variável “Resultado”

separadamente em cada país, pois caso seja encontrado comportamentos diferentes, pode não ser interessante unir os dados no momento da criação do modelo de Machine Learning.

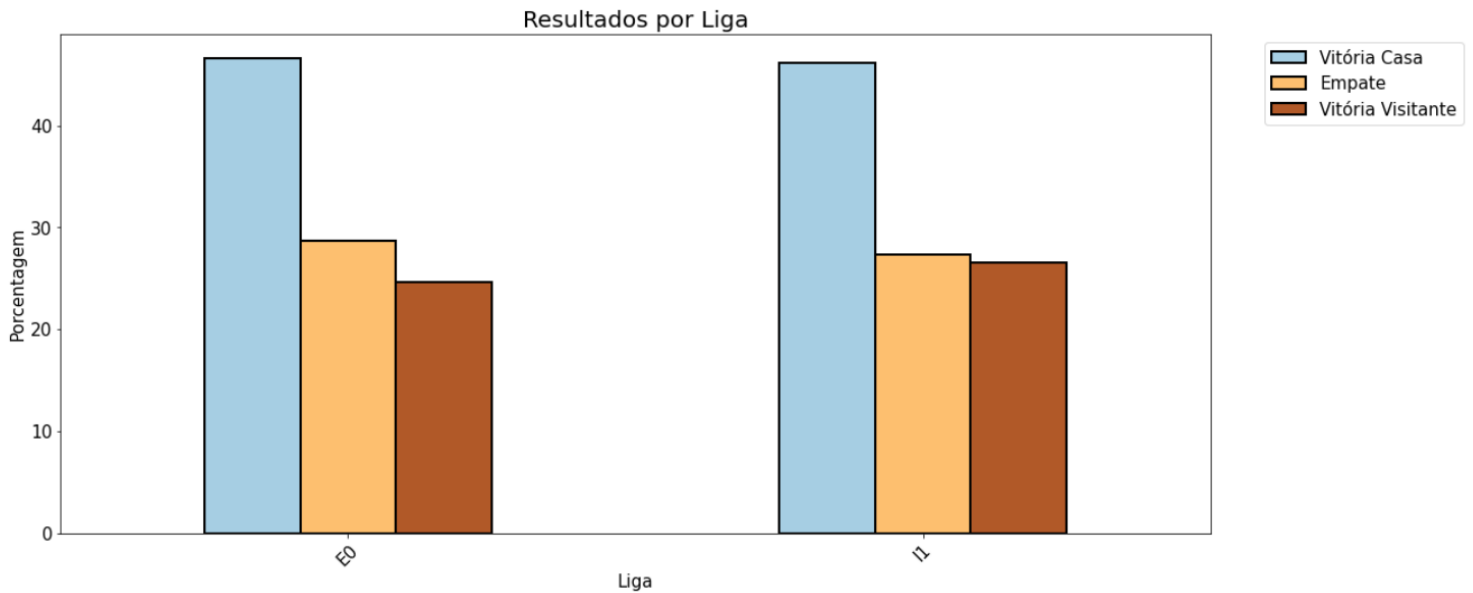


Figura 4.2 – Resultados por País. [E0 – Inglaterra, I1 – Itália]

O comportamento da coluna “Resultado” foi muito semelhante nos dois países, levando a conclusão que no momento da aplicação dos modelos de Machine Learning estes dados poderão ser trabalhados de forma conjunta.

As imagens abaixo reforçam a semelhança do comportamento dos “Resultados” mesmo em países diferentes.

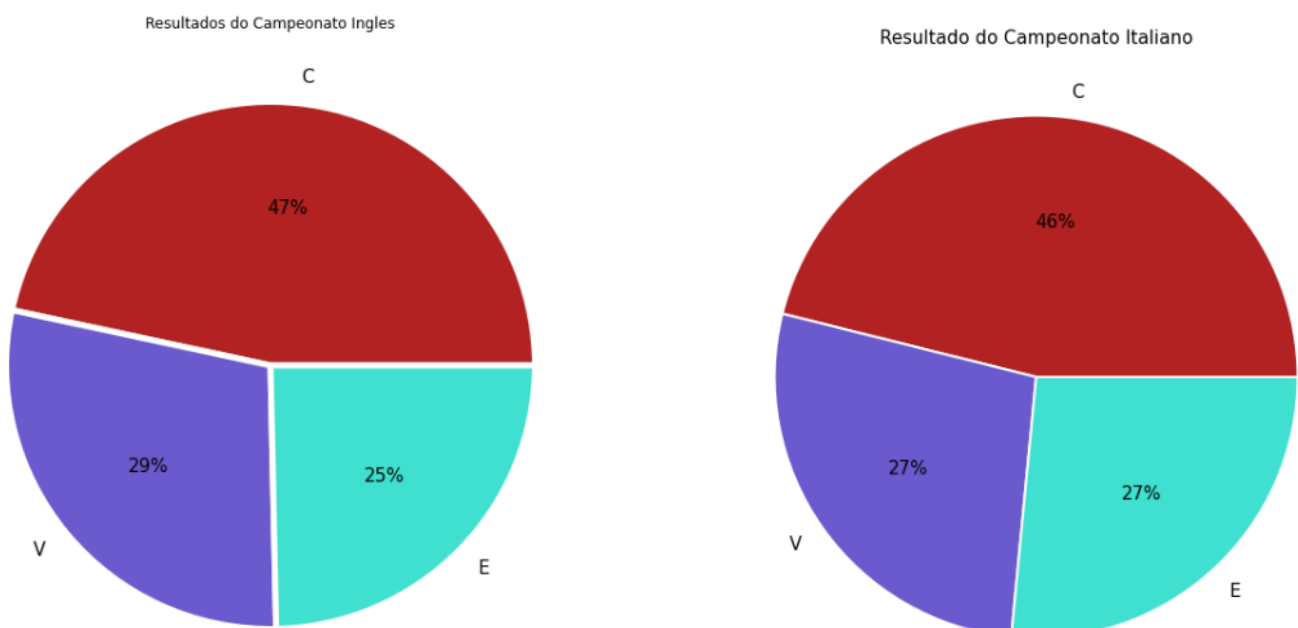


Figura 4.3 – Comparação de Resultados entre diferente países

## 5. Criação de Modelos de Machine Learning

Figura 5.1 – Importação de bibliotecas

O trabalho foi iniciado fazendo a importação do dataset que foi trabalhado na sessão 3 deste trabalho. Neste momento o dataset, agora chamado de “df” possui 10640 linhas e 38 colunas.

O primeiro passo foi eliminar as linhas onde a rodada fosse menor que 3, pois estes dados não possuíam o retrospecto anterior das equipes. Dessa forma o df ficou com 9800 linhas.

```
df = df.loc[df['Rodada'] > 3]
df.shape

(9800, 38)
```

Figura 5.2 – Eliminação das primeiras linhas do df

Em seguida foram eliminadas as colunas que não seriam necessárias nos modelos de Machine Learning. Foram eliminadas as colunas:

- Div
- Data
- Casa
- Visitante
- Placar Casa
- Placar Visitante
- Pontos Casa

O próximo passo foi a divisão dos resultados acumulados pela rodada em que o confronto estava ocorrendo. Desta forma os elevados valores de pontos e gols acumulados no fim do campeonato, não exercerão influência em alguns modelos de Machine Learning que serão utilizados.

```
df['Rodada'] = df['Rodada'].astype(float)

for colunas in escala:
    df[colunas] = df[colunas] / df['Rodada']
```

Figura 5.3 – Escala usando a coluna “Rodada” como referência

Também foi realizada a escala entre as variáveis através da biblioteca `sklearn.preprocessing`.

```
from sklearn.preprocessing import scale

for col in escala:
    x[col] = scale(x[col])
```

Figura 5.4 – Aplicação de scale

O passo seguinte foi trabalhar em cima das variáveis do tipo string, usando a função do pandas `get_dummies`. Foi trabalhado em cima das colunas: C1, C2, C3, V1, V2, V3, Casa Sequencia, Visitante Sequencia.

C1_D	C1_E	C1_S	C2_D	C2_E	C2_S	C3_D	C3_E	C3_S	V1_D	V1_E	V1_S	V2_D	V2_E	V2_S	V3_D	V3_E	V3_S	Casa Sequencia_DDD	Casa Sequencia_DDE
1	0	0	0	0	1	1	0	0	0	0	1	1	0	0	1	0	0	0	0
0	0	1	0	0	1	0	0	1	0	1	0	0	0	1	0	0	1	0	0
1	0	0	1	0	0	0	0	1	1	0	0	1	0	0	0	1	0	0	0
0	1	0	0	0	1	0	1	0	0	1	0	1	0	0	1	0	0	0	0
0	0	1	0	0	1	1	0	0	0	0	1	0	0	1	0	0	1	0	0

Figura 5.5 – Parte do DF com as novas colunas após `get_dummies`

Dessa forma, como DF já separado em x e y, ficou com a parte x com 9800 linhas e 92 colunas. Conforme Figura 5.6.

```
x.columns
```

```
Index(['Soma Pontos Casa', 'Soma Pontos Visitante', 'Gols Casa',
      'Gols Visitantes', 'Gols Conc. Casa', 'Gols Conc. Visitante', 'GMC',
      'GMV', 'GCC', 'GCV', 'Total P. Casa', 'Total P. Visit.', 'Casa_AA',
      'Visitante_AA', 'Dif Gols Casa', 'Dif Gols Visit',
      'Dif Gols Time Casas', 'Dif Gols Time Visitante', 'Dif Pontos',
      'Dif Posição', 'C1_D', 'C1_E', 'C1_S', 'C2_D', 'C2_E', 'C2_S', 'C3_D',
      'C3_E', 'C3_S', 'V1_D', 'V1_E', 'V1_S', 'V2_D', 'V2_E', 'V2_S', 'V3_D',
      'V3_E', 'V3_S', 'Casa Sequencia_DDD', 'Casa Sequencia_DDE',
      'Casa Sequencia_DDS', 'Casa Sequencia_DED', 'Casa Sequencia_DEE',
      'Casa Sequencia_DES', 'Casa Sequencia_DSD', 'Casa Sequencia_DSE',
      'Casa Sequencia_DSS', 'Casa Sequencia_EDD', 'Casa Sequencia_EDE',
      'Casa Sequencia_EDS', 'Casa Sequencia_EED', 'Casa Sequencia_EEE',
      'Casa Sequencia_EES', 'Casa Sequencia_ESD', 'Casa Sequencia_ESE',
      'Casa Sequencia_ESS', 'Casa Sequencia_SDD', 'Casa Sequencia_SDE',
      'Casa Sequencia_SDS', 'Casa Sequencia_SED', 'Casa Sequencia_SEE',
      'Casa Sequencia_SES', 'Casa Sequencia SSD', 'Casa Sequencia_SSE',
      'Casa Sequencia_SSS', 'Visitante Sequencia_DDD',
      'Visitante Sequencia_DDE', 'Visitante Sequencia_DDS',
      'Visitante Sequencia_DED', 'Visitante Sequencia_DEE',
      'Visitante Sequencia_DES', 'Visitante Sequencia_DSD',
      'Visitante Sequencia_DSE', 'Visitante Sequencia_DSS',
      'Visitante Sequencia_EDD', 'Visitante Sequencia_EDE',
      'Visitante Sequencia_EDS', 'Visitante Sequencia_EED',
      'Visitante Sequencia_EEE', 'Visitante Sequencia_EES',
      'Visitante Sequencia_ESD', 'Visitante Sequencia_ESE',
      'Visitante Sequencia_ESS', 'Visitante Sequencia_SDD',
      'Visitante Sequencia_SDE', 'Visitante Sequencia_SDS',
      'Visitante Sequencia_SED', 'Visitante Sequencia_SEE',
      'Visitante Sequencia_SES', 'Visitante Sequencia SSD',
      'Visitante Sequencia_SSE', 'Visitante Sequencia_SSS'],
      dtype='object')
```

Figura 5.6 – Todas as colunas para aplicação de Machine Learning

Como último passo foi feita a divisão dos dados utilizando a biblioteca Scikit-Learn, conforme mostrado na imagem abaixo:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(x, y,
                                                    test_size = 150,
                                                    random_state = 1,
                                                    stratify = y)
```

Figura 5.7 – Aplicação de train\_test\_split

Também foi feita uma manipulação nos dados da coluna Resultado, antes era possível encontrar 3 resultados distintos, vitória, derrota e empate. Mas foi feita a opção de transformar essa coluna em 1 (vitória da casa) e 0 (não vitória da casa).



## 6. Apresentação dos Resultados

Os dados foram submetidos a três modelos indicados para classificação; Regressão Logística, SVM e Ramdon Tree Forest.

Começando pelo modelo de Regressão Logística, foi feita a importação da biblioteca do Scikit-Learn, conforme as figuras abaixo. E a aplicação nos dados já separados em X\_train, X\_test, y\_train, y\_test.

```
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression()
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)

0.7133333333333334
```

Figura 6.1 – Aplicação de Logistic Regression

O modelo apresentou uma eficácia de 71,3%.

Fazendo uma análise dos dados “y\_test” é encontrado que a amostra possui 80 registros de valor 0 e 70 registros de valor 1. O que reforça que o modelo atingiu um resultado satisfatório.

```
y_test.value_counts()

0      80
1      70
Name: Resultado, dtype: int64
```

Figura 6.2 – Composição do y\_test

Fazendo uma análise da matriz de confusão foi possível perceber que o modelo se comportou bem na predição das duas categorias, acertando 60 dos 80 resultados 0, uma taxa de 75%. E acertou 47 dos 70 resultados 1, uma taxa de acerto de 67%.

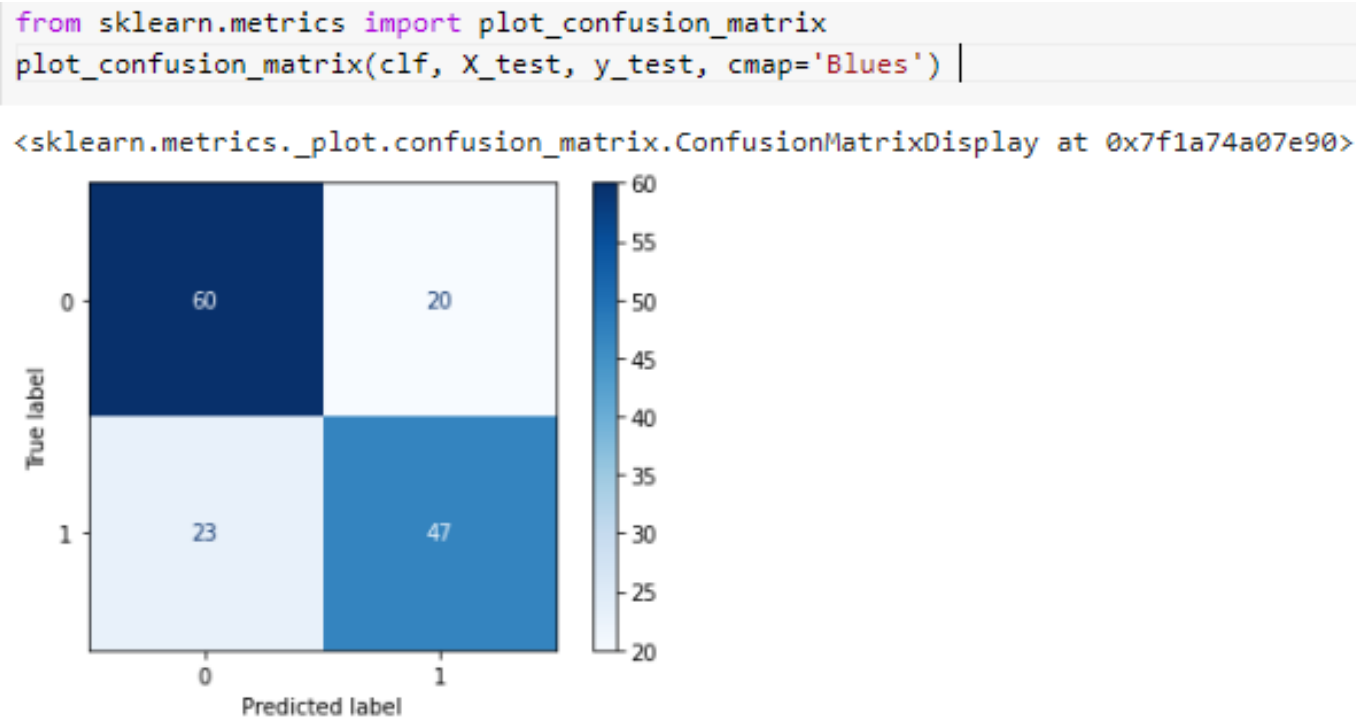


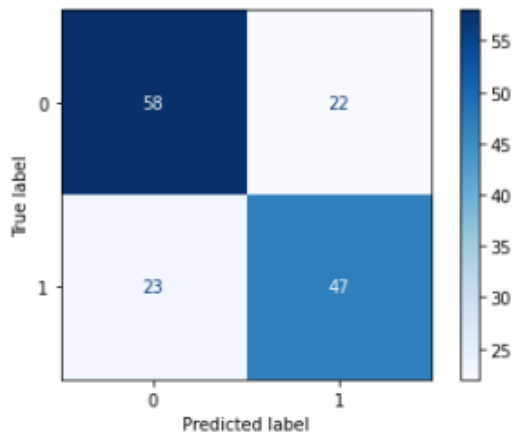
Figura 6.3 – Matriz de Confusão Regressão Logística

Os modelos SVM e Random Tree Forest apresentaram resultados inferiores quando comparados com a Regressão Logística, mas

O modelo SVM performou com o mesmo índice da Regressão Logística dentro da categoria 1 do `y_test`, porém teve performance inferior dentro da categoria 0.

```
47] from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(clf_svc, X_test, y_test, cmap='Blues')

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f1a74ffde90>
```



```
48] from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred2)

0.7
```

Figura 6.4 – Matriz de Confusão SVR

Já o modelo Random Tree Forest teve um desempenho ótimo na categoria 0, fazendo a previsão correta em 63 das 80 ocorrências. Um valor superior a 78% de acerto. Porém o modelo não se comportou bem dentro da categoria 1, acertando somente 38 vezes, contra 47 dos demais modelos. Isso totalizou uma acurácia de aproximadamente 67%.

```
| from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(clf_rfc, X_test, y_test, cmap='Blues')

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f1a7467c8d0>
```

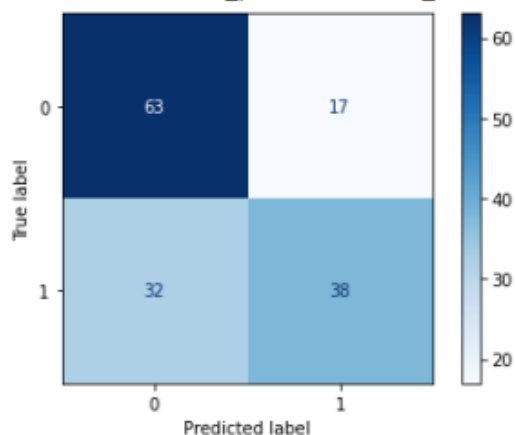


Figura 6.5 – Matriz de Confusão Random Tree Forest

Os modelos de previsão se mostraram satisfatórios, principalmente o modelo de Regressão Logística, com índices superiores a 70%.

Uma aplicação seria no crescente mercado de apostas esportivas, é possível desenvolver modelos que calculem qual o valor justo a se pagar em uma aposta.

Modelos de previsão de resultados esportivos podem ter diversas aplicações, tal como a aplicação direta por clubes de futebol. Hoje muitos clubes pelo mundo já possuem equipe especializada em Big Data e Machine Learning, para que junto da comissão técnica possa reverter aqueles jogos com baixa probabilidade de vitória.

## 7. Links

Link para o vídeo: <https://youtu.be/avT0nvx6hL0>

Link para o repositório: <https://github.com/pablofonseca14/projetotcc>