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

Fernando Berckembroch

MODELOS PREDITIVOS DO CAMPEONATO BRASILEIRO DE FUTEBOL DE 2014 A 2019

Belo Horizonte
2020

Fernando Berckembroch

MODELOS PREDITIVOS DO CAMPEONATO BRASILEIRO DE FUTEBOL DE 2014 A 2019

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
2020

SUMÁRIO

1. Introdução

1.1. Contextualização

Desde que Charles Miller chegou da Inglaterra com duas bolas e fundou o São Paulo Athletic Club, em 1894, o futebol foi conquistando os brasileiros, inicialmente a elite, posteriormente as classes populares. Hoje, o esporte é conhecido como uma paixão nacional e todos os seus resultados são amplamente divulgados nos meios de comunicação e mídias sociais, podendo assim ser acompanhados e estudados.

O Campeonato Brasileiro de Futebol começou a ser disputado em 1959, sob o nome de Taça Brasil, sendo que já recebeu vários nomes e formas de disputas. Em 1989 recebeu a denominação Campeonato Brasileiro de Futebol, e é usado até os dias atuais com exceção do ano 2000 quando foi denominado Copa João Havelange.

A partir de 2003, o Campeonato Brasileiro passou a ser disputado no sistema de pontos corridos, ou seja, um time joga contra todos os outros em turno e returno, e ao final, o maior acumulador de pontos é declarado campeão. Essa edição contou com 24 clubes, número que veio a diminuir em 2006, passando para 20, assim como é atualmente.

Em 2004, a rede Globo, canal de TV aberta, lançou o *fantasy game* Cartola FC, onde o usuário escala os jogadores da vida real, onze jogadores mais um técnico, que disputam o Campeonato Brasileiro e recebem pontuações conforme suas atuações, podendo ser tanto positivas quanto negativas.

Diante disto, é possível levantar as informações históricas de confrontos e resultados das equipes bem como as pontuações e preço dos jogadores no *fantasy game* Cartola FC. Nesse contexto, o trabalho visa utilizar técnicas de modelos preditivos para encontrar vencedores de jogos futuros com base em seu passado histórico.

1.2. O problema proposto

Com base nos dados históricos do Campeonato Brasileiro de Futebol da Serie A, entre os anos de 2014 e 2019 e com dados do *fantasy game* Cartola FC do mesmo período, foi criado indicadores para buscar identificar tendências para a previsão de resultados no Campeonato Brasileiro de Futebol. Por meio de classificadores utilizando *machine learning*, os dados foram coletados no site KAGGLE e no GITHUB utilizando *datasets* extraídos com ferramentas como *Python* e *Knime*.

Para isso tem-se como objetivos dessa análise:

- Analise exploratória dos *datasets;*
- Criar indicadores para analise preditiva;
- Criar modelo preditivo utilizando *Random Florest;*
- Criar modelo preditivo utilizando *Naive Bayes;*
- Criar modelo preditivo utilizando Redes Neurais(PNN);
- Criar modelo preditivo utilizando *Gradient Boosted Trees*;

Foi utilizado o período de jogos de 2014 até 2018 para analise exploratória e treinamento dos modelos de predição, e o ano de 2019 para teste e validação dos modelos.

2. Coleta de Dados

Os dados utilizados para este trabalho foram coletados por meio do site KAGGLE (kaggle.com), uma comunidade subsidiaria da GOOGLE, utilizada por cientistas de dados e por entusiastas de técnicas de aprendizagem de máquina, onde é possível publicar e encontrar *datasets.*

Também foram utilizados dados do GITHUB, (github.com), uma plataforma de compartilhamento de código, onde é possível um programador visualizar o código e dados de outro e contribuir nos projetos *open source.*

Mediante o KAGGLE (www.kaggle.com/adaoduque/campeonato-brasileiro-de-futebol) foi coletado o conjunto de dados com o seguinte nome: “campeonato-brasileiro-full.csv”, nele contém o seguinte formato:

Tabela 1: Estrutura *DataSet*

Coluna	Descrição	Tipo
Horário	Horário que a partida ocorreu	*Varchar*
Dia	Dia da semana que a partida ocorreu	*Varchar*
Data	Data que ocorreu a partida	*Date*
Clube 1	Clube mandante, ou seja, o que joga em casa.	*Varchar*
Clube 2	Clube visitante	*Varchar*
Vencedor	Clube vencedor da partida.	*Varchar*
Rodada	Rodada do campeonato que ocorreu a partida.	*Varchar*
Arena	Local, estádio onde ocorreu a partida.	*Varchar*
Clube 1 Gols	Gols que o clube mandante realizou na partida.	*Integer*

Clube 2 Gols	Gols que o clube visitante realizou na partida.	*Integer*
Clube 1 Estado	Estado do clube mandante.	*Varchar*
Clube 2 Estado	Estado do clube visitante.	*Varchar*
Estado Clube Vencedor	Estado do clube vencedor.	*Varchar*

Esse *dataset* possui um total de 7.940 linhas onde traz todos os confrontos desde o ano 2000. Para a análise, foram utilizados apenas os dados dos jogos de 2008 até o ano de 2019, totalizando 4.558 linhas. Já os dados de 2008 até 2013 serão utilizadas em alguns indicadores.

Pelo GITHUB (https://github.com/henriquepgomide/caRtola/tree/master/data) foi coletado o conjunto de dados denominado, “dados_agregados_limpos.csv”, que originalmente eram do *fantasy game* Cartola FC, e possui os seguintes campos:

Tabela 2: Estrutura *DataSet*

Coluna	Descrição	Tipo
Apelido	Apelido do Jogador	*Varchar*
AtletaID	Código único do jogador	*Integer*
ClubeID	Código do clube	*Integer*
Participou	Participou ou não do jogo	*Boolean*
Pontos	Número de pontos obtidos	*Numeric*
Preco	Preço do jogador na rodada do *fantasy game* Cartola FC	*Numeric*
Rodada	Número da rodada do campeonato	*Integer*
Ano	Ano da realização da partida	*Integer*
Dia	Dia da realização da partida	*Numeric*
Mês	Mês da realização da partida	*Numeric*

Esse conjunto de dados possui informações de 2014 até 2017. Para os anos de 2018 e 2019 foi utilizado outro *dataset*, também retirado do site GITHUB (https://github.com/henriquepgomide/caRtola/tree/master/data/2018), onde possuem vários arquivos com os nomes “rodada-*.csv”, por rodada do campeonato, ou seja, são trinta e oito arquivos por ano, isso vale também para 2019 (https://github.com/henriquepgomide/caRtola/tree/master/data/2019), pois a estrutura é a mesma, conforme abaixo:

Tabela 3: Estrutura *DataSet*

Coluna	Descrição	Tipo
Atletas.apelido	Apelido do Jogador	*Varchar*
Atletas.rodada_id	Número da rodada do campeonato	*Integer*
Atletas.pontos_num	Número de pontos obtidos	*Numeric*
Atletas.preco_num	Preço do jogador no *fantasy game* Cartola FC	*Numeric*
Atletas.clube.id.full.name	Nome do time do jogador	*Numeric*

3. Processamento/Tratamento de Dados

Inicialmente é preciso encontrar alguns indicadores para tentar chegar a um resultado minimamente satisfatório, na visão do autor 50% de acurácia. Para a criação desses indicadores foi processado o *dataset* “campeonato-brasileiro-full.csv” utilizando o *Python* e alguma de suas bibliotecas da seguinte forma:

Em seguida, foi importado o *dataset* e feito alguns tratamentos, filtrado apenas os jogos de 2013 em diante. Foi feito isso para a criação de alguns indicadores possuírem dados anteriores à primeira partida da análise que seria em 2014.

A ação ocorreu da seguinte forma:

Figura 1 – Print código em *python*

```
from scipy.stats import poisson
import pandas as pd
import numpy as np
import math

data = pd.read_csv("D:\DATASETS\campeonato-brasileiro-full.csv",sep=',')
df = pd.DataFrame(data)
df_pc2013 = df[df['Data'] >= '2013-01-01 00:00:00']
jogos = df_pc2013.copy()
```

Quando ocorre um empate, o *dataset* retornava apenas um caracter “-“ na coluna “Vencedor”. Para contornar essa questão, foi substituído pela palavra “empate”, também para padronizar os campos “Clube 1”, “Clube 2” e a própria coluna “Vencedor”, deixando todas com letra minúscula:

Figura 2 – Print código em *python*

```
jogos['Clube 1'] = jogos['Clube 1'].astype(str).str.lower()
jogos['Clube 2'] = jogos['Clube 2'].astype(str).str.lower()
jogos['Vencedor'] = jogos['Vencedor'].astype(str).str.lower()
jogos['Vencedor'] = jogos['Vencedor'].replace('-','empate')
```

a) Razão de Vitória:

Para calcular esse indicador, utilizamos os resultados dos times até a data do jogo, ou seja, para as partidas iniciadas em 2014, foram utilizados os dados desde o início de 2013. Nesse indicador para os times que não participaram dos jogos de 2013, irá retornar zero, para os demais o cálculo feito foi o seguinte:

Primeiro encontramos o número de vitórias desde o início de 2013 até a data atual do jogo, depois foi dividido pelo número de partidas jogadas pelo time nesse mesmo período.

Para encontrar o número de vitórias foi utilizada a função:

Figura 3 – Print código em *python*

```python
def getVitoriasTotalAteOJogo(dados, time, datajogo):
    filter1 = dados["Vencedor"].str.lower() == time
    filter2 = dados["Data"] < datajogo
    df = dados[((filter1) & (filter2))]
    vitoria = df['Vencedor'].count()
    return vitoria.astype(np.int64)
```

Onde o primeiro parâmetro seria o *dataset*; o segundo, o time que está sendo analisado, e, por fim, a data da partida, que retorna o número de vitórias.

Para encontrar o número de jogos, a função abaixo possui os mesmos parâmetros de entrada e retorna o número de partidas até a data do jogo:

Figura 4 – Print código em *python*

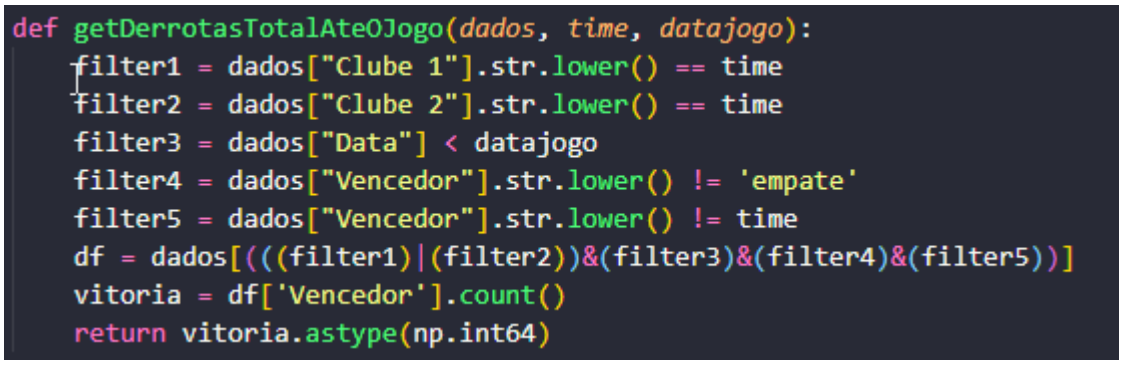
```python
def getJogosAteOJogo(dados, time, datajogo):
    filter1 = dados["Clube 1"].str.lower() == time
    filter2 = dados["Clube 2"].str.lower() == time
    filter3 = dados["Data"] < datajogo
    return (dados[(filter1) & (filter3)]["Clube 1"].count() + dados[(filter2) & (filter3)]["Clube 2"].count()).astype(np.int64)
```

Após a obtenção desses dados, foi feito a divisão do número de vitórias pelo total de jogos, chegando assim no resultado por time para a partida.

b) Razão de Derrota:

Esse indicador assemelha-se praticamente em tudo ao razão de vitória, porém, trata-se da razão de derrotas, onde também é utilizado a função “getJogosAteOJogo” para ser utilizado no divisor. É utilizado para a obtenção do número de derrotas a função:

Figura 5 – Print código em *python*



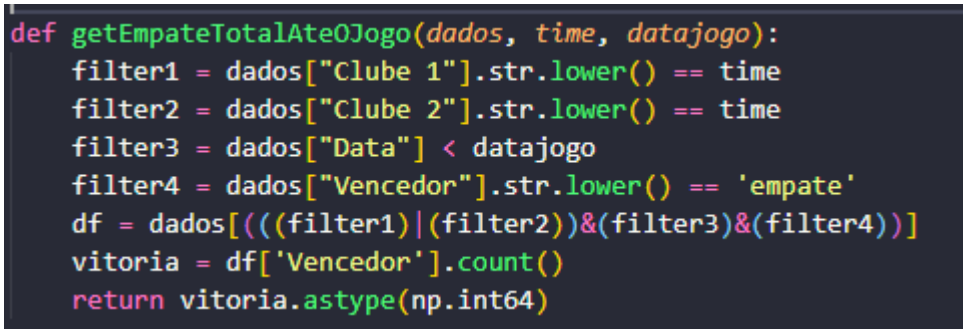
```python
def getDerrotasTotalAteOJogo(dados, time, datajogo):
    filter1 = dados["Clube 1"].str.lower() == time
    filter2 = dados["Clube 2"].str.lower() == time
    filter3 = dados["Data"] < datajogo
    filter4 = dados["Vencedor"].str.lower() != 'empate'
    filter5 = dados["Vencedor"].str.lower() != time
    df = dados[(((filter1)|(filter2))&(filter3)&(filter4)&(filter5))]
    vitoria = df['Vencedor'].count()
    return vitoria.astype(np.int64)
```

A exemplo do indicador anterior, é dividido o número de derrotas pelo total de jogos, chegando assim no valor do indicador.

c) Razão de Empate:

Para o cálculo desse indicador, foi utilizado novamente a função “getJogosAteOJogo” para ser utilizado no divisor. Para a obtenção do número de empates foi utilizado a função:

Figura 6 – Print código em *python*



```python
def getEmpateTotalAteOJogo(dados, time, datajogo):
    filter1 = dados["Clube 1"].str.lower() == time
    filter2 = dados["Clube 2"].str.lower() == time
    filter3 = dados["Data"] < datajogo
    filter4 = dados["Vencedor"].str.lower() == 'empate'
    df = dados[(((filter1)|(filter2))&(filter3)&(filter4))]
    vitoria = df['Vencedor'].count()
    return vitoria.astype(np.int64)
```

Conforme as funções anteriores tem por parâmetro apenas o *dataset*, o time que está sendo analisado e a data do jogo, retornando o número de empates.

d) Média de Gols:

A média de gols foi calculada do início de 2013 até a data da realização da partida, nos moldes dos indicadores de vitória/derrota/empate, para isso foi utilizado a função:

Figura 7 – Print código em *python*

```python
def getGpGeral (dados, time, datajogo):
    filter1 = dados["Clube 1"].str.lower() == time
    filter2 = dados["Clube 2"].str.lower() == time
    filter3 = dados["Data"] < datajogo
    df1 = dados[(filter3) & ((filter2) | (filter1))]


    gp = 0
    gc = 0

    for index, row in df1.iterrows():
        if (row["Clube 1"] == time):
            golsPro = row['Clube 1 Gols']
            golsCtr = row['Clube 2 Gols']
        else:
            golsPro = row['Clube 2 Gols']
            golsCtr = row['Clube 1 Gols']
        gp = (gp + golsPro)
        gc = (gc + golsCtr)
    return gp, gc
```

Nela passamos por parâmetro o *dataset*, time e data da partida, e ela nos retorna os gols realizados e sofridos até a data do jogo. Assim, para obter o resultado do nosso indicador, basta dividir os gols realizados pelo total de partidas, esse foi obtido também pela função “getJogosAteOJogo”.

e) Média de Gols Sofridos:

A criação desse indicador assemelha-se ao anterior, ao invés de usar o retorno gols realizados, foi utilizado o de gols sofridos e dividido pelo número de partidas.

f) Expectativa de Gols:

Esse é um dos indicadores mais complexos realizados nesse trabalho, pois para a criação do mesmo foi necessário vários cálculos para chegar a um valor.

O Cálculo realizado leva em conta o número de vitórias de mandantes versus o número de vitória de visitantes, em consideração aos times envolvidos na partida.

Inicialmente é calculada a força ofensiva do time i:

Fórmula 1 – Força ofensiva mandante

$$Alfa(i) = \frac{\frac{GM(i)}{TPM(i)}}{\frac{TGM}{TP}}$$

Onde GM, corresponde ao total de gols do time como mandante, TPM, ao total de partidas como mandante, TGM total de gols de todos os mandantes até a data da partida, e TP, total de partidas realizadas até o momento.

Após o cálculo da força ofensiva do primeiro time, aqui definidada como mandante, calculamos a força ofensiva do segundo time, o visitante, atráves do cálculo semelhante:

Fórmula 2 – Força ofensiva visitante

$$Beta(j) = \frac{\frac{GV(j)}{TPV(j)}}{\frac{TGV}{TP}}$$

Assim o GV, seria o total de gols do time como visitantes, TPV, o total de partidas como visitantes, TGV, total de gols de todos os visitantes até o momento, e TP, o total de partidas realizadas até o momento.

Os próximos passos calcularam a força defensiva tanto do mandante (Ômega) quando do visitante (Gama), com fórmulas semelhantes conforme abaixo:

Fórmula 3 – Força defensiva mandante

$$Ômega(i) = \frac{\frac{GSM(i)}{TPM(i)}}{\frac{TGSM}{TP}}$$

Fórmula 4 – Força defensiva visitante

$$Gama(j) = \frac{\frac{GSV(j)}{TPV(j)}}{\frac{TGSV}{TP}}$$

Onde o “S” da fórmula seria os gols sofridos, ou seja, inicialmente tratamos dos gols feitos e para calcular a força defensiva foi utilizado o número de gols sofridos.

Após o cálculo dessas quatro variáveis, e chegado em dois valores auxiliares um para o mandante e outro para o visitante, através do cálculo:

Fórmula 5 – Cálculo dos auxiliares

$$EGm(i) = Alfa * Beta * (TGM/TP)$$

$$EGv(j) = Gama * Omega * (TGM/TP)$$

Para finalizar, os valores das auxiliares EGm(i) e EGv(j) são submetidos a distribuição de probabilidade de Poisson para k e n gols para cada um dos auxiliares, em nosso estudo delimitamos valores entre zero e sete gols, chegando assim ao valor do nosso indicador, a expectativa de gols tanto para o mandante quanto para o visitante.

g) Forma:

Esse indicador mostra a desempenho dos times, tanto mandante como visitante, em um período curto para refletir o momento do clube. Para realizar o cálculo, retornamos a três rodadas anteriores, caso o clube tenha ganhado é atribuído três pontos, empate um ponto, derrota zero ponto. O próximo cálculo é feito sob as duas rodadas anteriores, utilizando a mesma lógica, porém, com o resultado obtido multiplicado por dois, ou seja, peso dobrado.

A rodada anterior tem peso três, é identificada a pontuação do clube e multiplicado por três, após esses passo é somado à pontuação do time e chegado ao valor desse indicador.

h) Elo:

Calcular o ELO, é refletir o longo prazo o desempenho do time, ou seja, como ele se comportou nos últimos anos no Campeonato Brasileiro.

Para isso, foi calculado cinco anos anteriores ao início de nossa análise (2014), sempre um ano por vez para conseguir compor um valor inicial, assim, ao iniciar a análise já pudemos refletir o passado.

Inicialmente em 2008 foi atribuído o valor de 1.000 pontos para todos os clubes, a cada rodada é realizado um cálculo e atribuído um novo valor para cada time, conforme fórmula abaixo:

Fórmula 6 – Cálculo do ELO

$$nElo = oElo + G * (Wm - Wpm)$$

Onde o nElo seguinifica o novo elo e é calculado recebendo o elo atua, mais o valor de G que é um cálculo feito referente a diferença de gols entre os clubes no confronto. Quando a diferença é de 1 o valor de G é 1 quando o valor da diferença é 2 o valor de G é 1,5, acima dessa diferença é feito o cálculo:

Fórmula 7 – Cálculo do G

$$G = (dGols + 11) / 8$$

O valor de G é usado tanto para o mandante quando para o visitante, considerando a diferença de gols.

O Wpm é a expectativa de vitória, quando for favoravél o valor de Wpm é 1, e se for de derrota o valor de Wpm é 0, e para caso de empate recebe o valor 0.5.

Enquanto o Wpm é a expectativa de resultado, o Wm, é o resultado que realmente ocorreu, recebendo a mesma pontuação.

Assim o valor de G é multiplicado conforme a realização ou não da expectativa de vitória.

A maior dificuldade nesse passo é para os clubes rebaixados e para os clubes promovidos, pois nessa situação não teriam valores iniciais para o elo. Para contornar essa situação, foi atribuido a média dos quatro clubes rebaixados mais o primeiro fora da zona de rebaixamento, assim não penalizando o clube promovido.

Outra situação foi o hiper-inflacionamento do elo. Para contornar essa situação, ao final de cada cálculo dos anos, foi aplicado uma fórmula para compor o valor do elo inicial, onde 80% do valor seria do elo e outros 20% manteria o valor incial, ou seja, 200.

Após o cálculo do elo inicial, a partir de 2013, foi implementado os valores encontrados por clubes e calculado simultaneamente para os demais anos, sem compor ano a ano. O valor para os clubes promovidos foi atribuído uma média dos últimos quatro clubes rebaixados mais o primeiro fora da zona de descesso, dos cincos anos anteriores.

i) Expectativa de Vitória:

Para o cálculo da expectativa de vitória, usamos a expectativa de gols, com a mesma metodologia utilizada no item “F – Expectativa de gols”, com base nos seus valores é comparado os dados do clube mandante e visitante, quando o valor do mandante é maior que o do visitante atribuímos “M” para essa variável, quando é o contrário atribuímos “V”, e em caso dos valores serem iguais atribuímos “E”, de empate.

j) Preço:

O preço é um indicador retirado dos dados do *fantasy game* Cartola F.C. inicialmente foi importado o *dataset*, em um banco de dados POSTGRESQL, na versão 9.6, utilizando as funções:

Figura 8 – Print código em *python*

```
def ProcessaRegistros(pArquivo,pTabela,pCaminhoAuxiliar):
    try:
        connection = psycopg2.connect("host=localhost port='5434' dbname=tcc_fernando user=fernando password=[redacted]")
        cursor = connection.cursor()
        csvfile = pd.read_csv(pArquivo ,sep=',',usecols = ['Apelido','AtletaID','ClubeID','Participou','Pontos','Preco',
                                                           'Rodada','ano','dia','mes'])

        #print(csvfile)
        csvfile.to_csv(pCaminhoAuxiliar +lvBarra + 'alterado.csv', index = False, sep = ',')

        csvImport = open(pCaminhoAuxiliar +lvBarra + 'alterado.csv',encoding='utf-8')
        next(csvImport)

        cursor.copy_from(csvImport, pTabela, sep=',', columns = ('Apelido','AtletaID','ClubeID','Participou','Pontos',
                                                                 'Preco','Rodada','Ano','Dia','Mes'))

        connection.commit()
    except (Exception, psycopg2.Error) as error :
        print ("Erro: ", error)
    finally:
        cursor.close()
        connection.close()
        print('Registros inseridos com sucesso '+str(datetime.now()))
```

Após montar os dados para inserção, é percorrido a pasta onde os arquivos estão para que seja possível ler todos de uma vez.

Figura 9 – Print código em *python*

```python
def MontarArquivos(pCaminhoArquivo,pTabela,pCaminhoAuxiliar):
    list_files = os.listdir(pCaminhoArquivo)

    for arquivo in list_files:
        lvIniTime = str(datetime.now())
        lvArquivo = pCaminhoArquivo  +lvBarra+  arquivo;
        print('Lendo arquivo ['+lvIniTime+'] ' + lvArquivo)

        ProcessaRegistros(lvArquivo,pTabela,pCaminhoAuxiliar)


        lvFimTime = lvIniTime = str(datetime.now())
        print('Fim arquivo ['+lvFimTime+'] ' + lvArquivo)
        print('**********************************************************************************************************')

MontarArquivos(lvDirArquivo,lvTable,lvDirArquivoAux);
```

Após a importação no banco de dados, foi agrupado em uma tabela os valores de preço e pontos por time, mediante uma consulta SQL conforme abaixo:

Figura 10 – Print código em SQL

```sql
DROP TABLE IF EXISTS forca_times_cartola;
select * INTO forca_times_cartola FROM (

SELECT LOWER(clubeid) AS clube,
    ano,
    rodada,
    SUM(pontos) AS pontos,
    SUM(preco) AS preco
FROM imp_cartola
GROUP BY 1,2,3
UNION ALL
SELECT LOWER(clube) AS clube,
    2018::INTEGER AS ano,
    rodada,
    SUM(pontos) AS pontos,
    SUM(preco) AS preco
from imp_cartola_2
GROUP BY 1,3
UNION ALL
SELECT LOWER(clube) AS clube,
    2019::INTEGER AS ano,
    rodada,
    SUM(pontos) AS pontos,
    SUM(preco) AS preco
FROM imp_cartola_3
GROUP BY 1,3
ORDER BY 1,2,3
) AS A;
```

Após a criação da tabela “forca_times_cartola”, foi necessário fazer alguns tratamentos, como, por exemplo, dados faltantes de algumas rodadas para alguns times. Nessa situação foi concluído que o melhor seria repetir os valores da rodada anterior, lembrando que os valores para a rodada em análise são os obtidos na rodada anterior. utilizada da rodada anterior.

k) Pontos:

Da mesma forma que no item anterior o indicador Pontos foi criado levado em consideração os dados do *fantasy game* Cartola FC, com a mesma metodologia, somando agora a pontuação dos jogadores da rodada, agrupando pelo time e pela própria rodada.

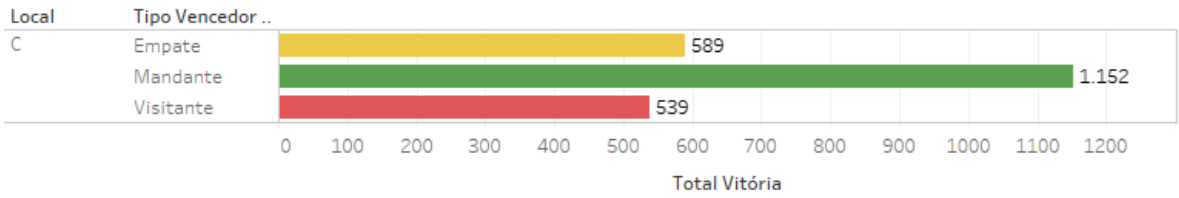
Ao final do cálculo de todos os indicadores, tanto os que foram feitos via *python,* quanto os que foram somados no *postgresql*, foram inseridos em uma única tabela no *postgresql* juntados via script SQL e criado um novo *dataset*, com um total de 30 colunas por 2.280 linhas, ou seja, um total de 380 jogos por ano.

4. Análise e Exploração dos Dados

Inicialmente os indicadores criados levavam em consideração apenas os resultados mais recentes dos times, como pontos e gols dos últimos 10 e 5 jogos, porém, foi percebido que esses padrões não conseguiam refletir a realidade da partida, por isso, o projeto foi todo repensando para mostrasse a “vida” do time antes daquele jogo.

Na análise exploratória foi identificado que a maioria dos vencedores são os mandantes, ou seja, o fato de um time jogar em casa parece ser um dos principais elementos para definir o vencedor da partida.

Gráfico 1 – Vitórias por tipo nos anos do estudo



O fator casa também está relacionado ao o número de gols, pois sempre que um time joga no seu estadio faz mais gols do que quando joga fora, reforçando assim a teoria de que o mandante tem vantagens.

5. Criação de Modelos de Machine Learning

Para a etapa de criação de modelos de *Machine Learning*, foi aplicada a ferramenta Knime, a partir do *dataset* criado com todos os indicadores tabulados. Para isso, foi utilizado algoritmos de classificação e previsão supervisionada, ou seja, quando buscamos prever uma variável dependente a partir de uma lista de variáveis independentes.

Foi enviada uma lista de indicadores para os classificadores para que pudessem fazer a melhor predição. Para todos foram às mesmas colunas, são elas: razão de vitória mandante e visitante, razão de derrota mandante e visitante, razão de empate mandante e visitante, média de gols mandante e visitante, média de gols sofridos de ambos os times, expectativa de gols mandante e visitante, elo mandante e visitante, expectativa de vitória mandante e visitante, assim como preço e ponto, tanto de mandante como de visitante.

Para efeito de comparação entre os modelos, utilizamos a medida da *Accuracy* (acurácia) com a seguinte fórmula:

Fórmula 8 – Accuracy

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

Simplificando: seria o total de acertos divididos pelo total, onde TP significa *True Positive*, TN *True Negative*, FP *False Positive*, FN *False Negative*. Também foi utilizado o F1-Score por unir as métricas de *Precision* e *Racall*. Assim, para realizar o cálculo precisamos saber primeiro a *Precision*, onde chegamos na seguinte fórmula:

Fórmula 9 – Precision

$$\text{Precision} = \frac{TP}{TP + FP}$$

Dividimos os *True Positive* pela soma do *True Positive* com os *False Positive*, chegando assim na precisão do modelo, e de forma similiar calculamos o Racall:

Fórmula 10 – Recall

$$Recall = \frac{TP}{TP+FN}$$

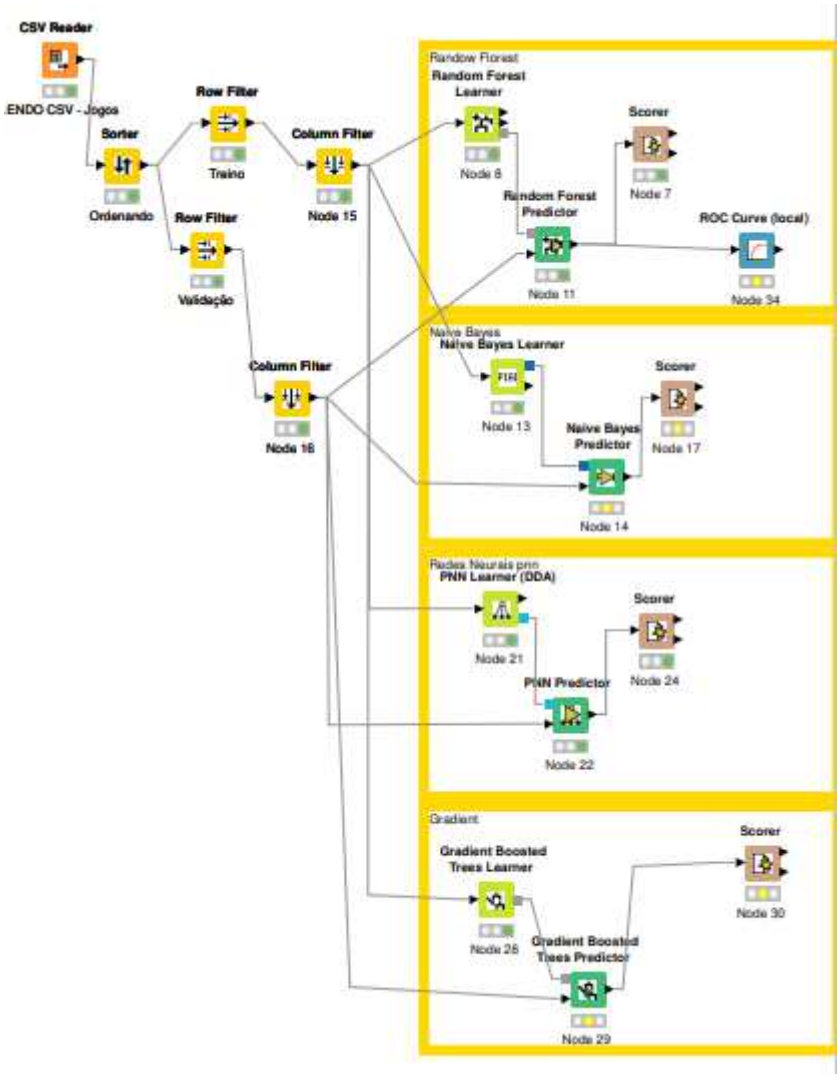
Porém, somamos os *False Negative* no divisor, assim com ambos os valores podemos aplicar a fórmula do F1-Score, que é utilizado para unir as duas medidas:

Fórmula 11 – F1-Score

$$F1\text{-}Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

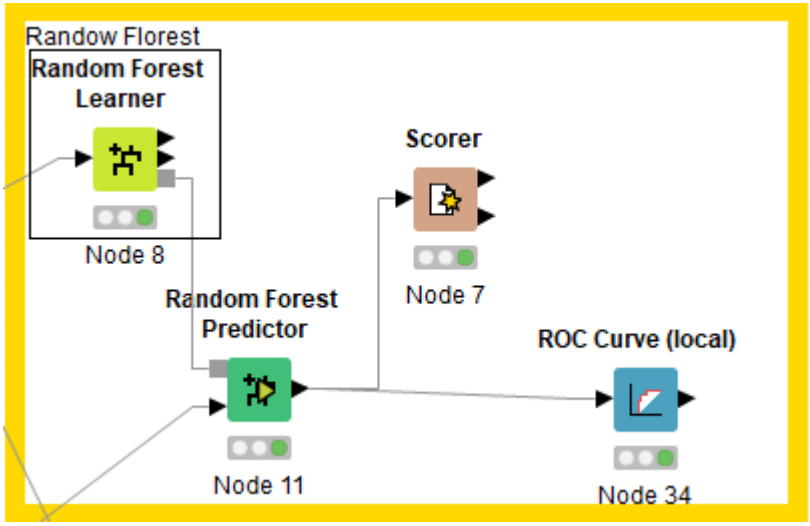
Abaixo o workflow completo criado na ferramenta Knime:

Figura 11 – Workflow Knime



O primeiro classificador utilizado foi o *Random Florest* (Floresta Aleatória). Nesse tipo de classificador, montam várias árvores de decisões e as percorrem encontrando os melhores indicadores a serem usados. Abaixo o workflow desse modelo:

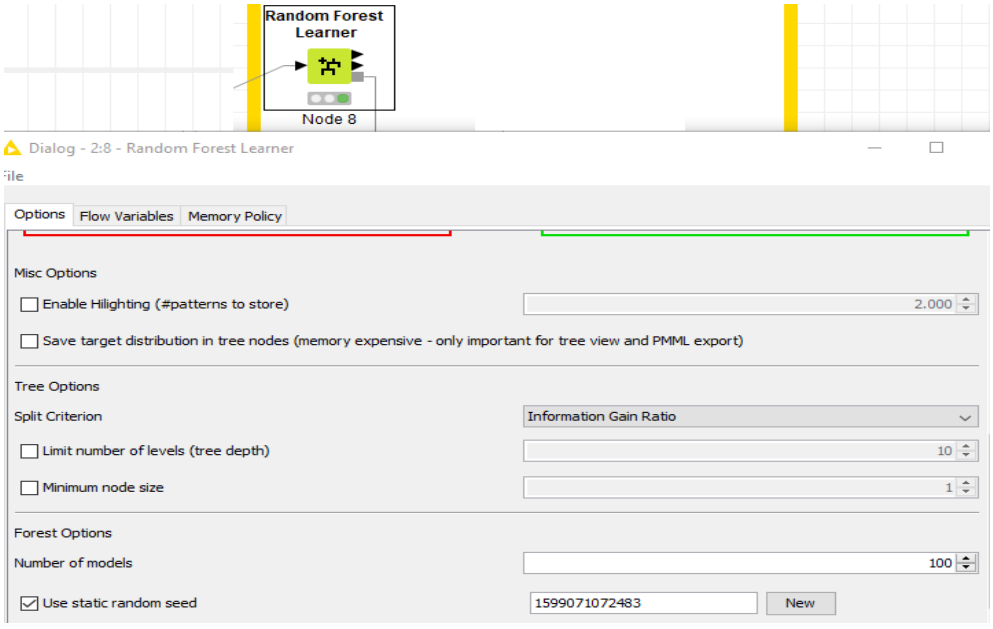
Figura 12 – Workflow Knime



Onde na parte de superior é recebido os valores de treinamento, e na parte da predição os valores de validação.

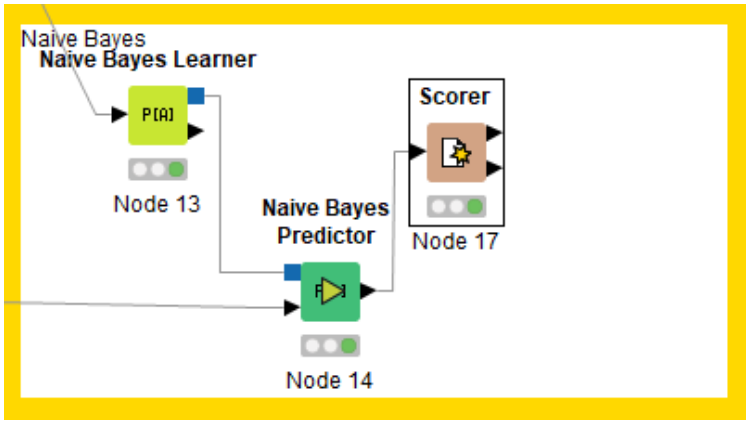
O parâmetro para esse classificador utilizou para o critério de divisão, é a razão de ganho e geração de 100 modelos.

Figura 13 – Configuração Random Florest Learner



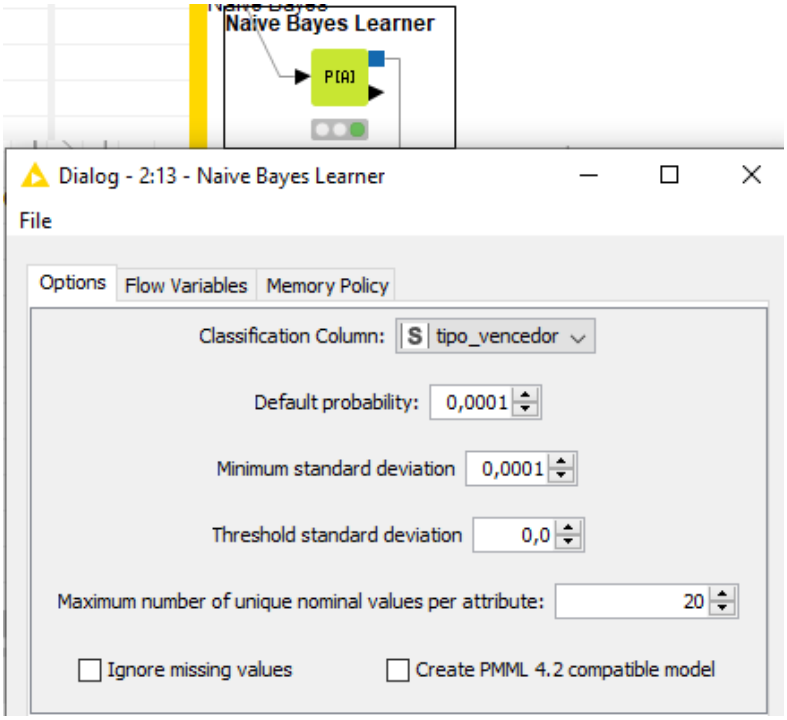
O segundo classificador utilizado foi o *Naive Bayes*, baseado no Teorema de Bayes, que foi criado por Thomas Bayes para tentar provar a existência de Deus. Sua característica é não considerar a correlação entre as variáveis, daí o “Naive” no seu nome, que significa ingênuo, ele compara as probabilidades chegando a um valor. Abaixo o workflow do modelo.

Figura 14 – Workflow Knime



Nesse indicador utilizamos sua forma *Gaussian*, ou seja, de distribuição normal, probabilidade padrão de 0,0001, desvio padrão mínimo de igual valor, e desvio padrão limite de 0,0 e número máximo de valores nominais por atributo em 20 conforme figura abaixo.

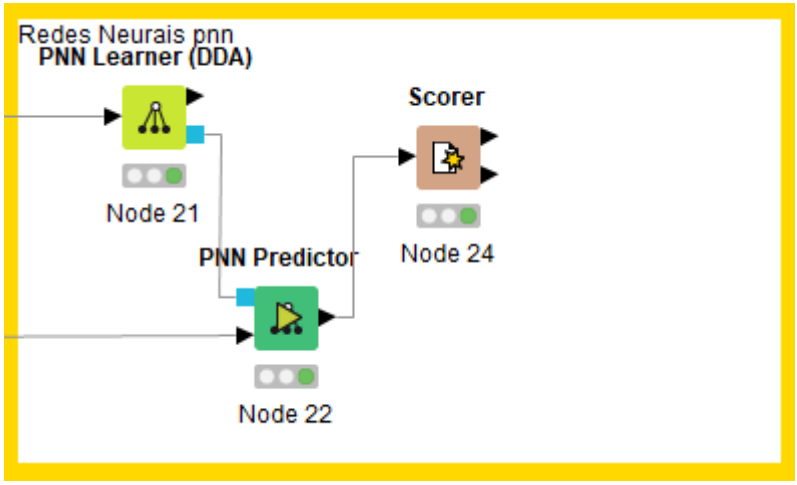
Figura 15 – Configuração parâmetros Naive Bayes



O terceiro classificador utilizado foi o de Redes Neurais Artificiais, que trabalha com nós integrados simulando um cérebro humano, podendo reconhecer padrões e correlações em dados brutos.

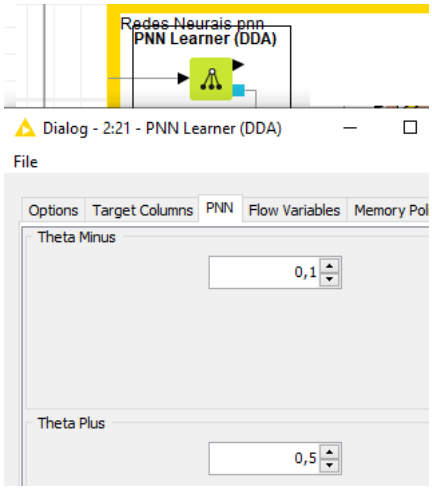
A forma de uso das Redes Neurais foi a Rede Neural Probabilística (Probabilistic Neural Network – PNN), com regra definida com uma função Gaussiana, e ajustada por dois limitantes (*Theta Minus* e *Theta Plus*) para evitar conflitos com regras de classes diferentes, conforme workflow abaixo.

Figura 16 – Workflow Knime



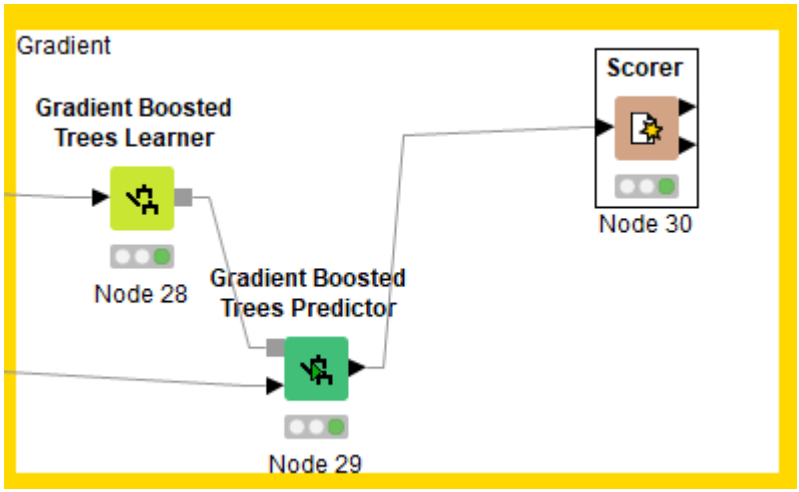
Os parâmetros usados para esse modelo para *Theta Minus* de 0,1 e para *Theta Plus* de 0,5 valores esses que aumentaram a acurácia do modelo.

Figura 17 – Configuração Redes Neurais (PNN)



Nosso quarto classificador trata-se do *Gradient Boosted,* semelhante ao *Random Florest,* por também utilizar árvores de decisão. Suas diferenças são as formas de construção das árvores e como é feito a combinação de resultado, abaixo workflow do modelo.

Figura 18 – Workflow Knime



Para esse modelo usamos como parâmetros o número de árvores em oito, o número máximo de modelos em 100 e a taxa de aprendizagem em 0,1, foi assim que chegamos às melhores previsões, conforme figura abaixo.

Figura 19 – Configuração Gradient Boosted

6. Apresentação dos Resultados

Inicialmente será apresentado o preenchimento do workflow motivador desse trabalho seguindo o modelo de canvas proposto por Vasandani, (pode ser visto aqui).

Figura 20 – Canvas projeto

Título: MODELOS PREDITIVOS DO CAMPEONATO BRASILEIRO DE FUTEBOL DE 2014 A 2019		
Definição do problema: Buscar identificar tendência para a previsão de resultados no Campeonato Brasileiro de futebol atráves de classificadores utilizando Machine Learning	**Resultados e Previsões:** No inicio do projeto buscava chegar pelo menos acima dos 50% de acurácia nos modelos, no primeiro momento não consegui, após a troca dos indicadores, todos os classificadores ficaram acima dos 60% de Acurácia.	**Aquisição dos Dados:** Os dados são oriundos de fontes abertas, disponíves e não pagas, sendo elas GitHub, e o Kaggle.
Modelagem: Inicialmente foi tratado o dataset retirado do Kaggle, criando indicadores por jogos, sempre com o valor para o mandate e para o visitante, depois foi feito o tratamento dos dados do Cartola (Retirados do GitHub), e unindo em um único dataset para a analise exploratória e para a leitura dos classificadores.	**Avaliação do Modelo:** Os modelos foram avaliados principalmente com duas métricas: Accuracy e F1-Score, principalmente pois buscamos o maior número de resultados corretos para as partidas.	**Preparação dos Dados:** Foi inicialmente criado indicadores, que buscam os dados históricos dos clubes e seus desempenhos em determinada situação, como jogando em casa ou fora, além de seu desempenho atual, comparado com os demais times do campeonato na mesma situação.

Como utilizamos quatro classificadores, vamos primeiramente apurar o resultado de cada um individualmente para depois conseguir comparar e encontrar o mais adequado para nossa análise.

Random Florest, esse classificador atingiu uma *Accuracy* de 0,7737 com mesmo valor para o F1-Score, como o alvo do estudo é o acerto das partidas, (*True Positive*), o modelo previu corretamente 294 partidas no universo de 380, lembrando que treinamos o modelo com os dados de 2014 a 2018 e validamos com os jogos de 2019.

Em uma análise mais a fundo nesse classificador é possível verificar que o maior índice de acertos foram nos clubes mandantes, conforme matriz de confusão abaixo, onde o “M” Significa mandante, “V” visitante e “E” empate;

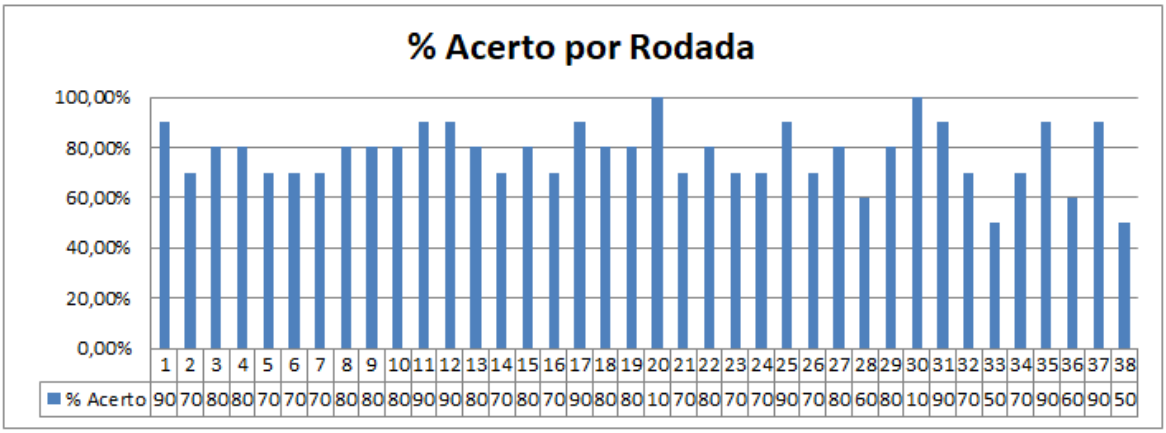
Figura 21 – Matriz de Confusão

Row ID	M	V	E
M	165	0	19
V	2	68	28
E	30	7	61

Assim, o modelo classificou corretamente 165 de 184 mandantes como vencedores (89,67%), para os visitantes o percentual ficou em 69,39%, enquanto o empate teve o pior desempenho de classificação, atingindo 62,24%. Esses dados nos monstram que para esse classificador foi mais fácil identificar os vencedores quando esses jogam em seus domínios, enquanto foi mais difícil verificar os empates.

Analisando rodada por rodada do ano 2019, que foi utilizada para validação, percebemos que a taxa de acerto na classificação cai nas últimas rodadas, principalmente após a trigésima terceira rodada. Isso ocorreu pois alguns times já não tinham mais pretenções no campenato e jogavam com um time alternativo, além disso, essa rodada, em especial, teve um total de seis empates e o modelo não conseguir classificar tão bem nesse cenário.

Gráfico 2 - % Acerto por Rodada



A contra ponto, na vigésima e na trigésima rodada o modelo conseguiu classificar corretamente 100% dos jogos. A vigésima rodada, em especial, teve 5 vitórias do mandante, 4 vitórias do visitante e apenas 1 empate, o que pode ter sido propício para o desempenho do modelo nessa rodada. Já na trigésima rodada, o modelo conseguiu classificar corretamente 3 partidas que tiveram empate.

Analisando por time, esse modelo conseguiu classificar corretamente 89,47% do time Ceará, ou seja, dos 38 jogos classificou corretamente 34, enquanto a pior classificação ficou por conta do time Palmeiras com 65,79%, 25 das 38 partidas disputadas, conforme tabela abaixo:

Tabela 5 – Lista de acertos por times

Time	% acerto
ceará	89,47%
cruzeiro	86,84%
botafogo-rj	84,21%
csa	84,21%
athlético-pr	81,58%
flamengo	81,58%
fortaleza	81,58%
fluminense	78,95%
internacional	78,95%
avaí	76,32%
chapecoense	76,32%
grêmio	76,32%
santos	76,32%
atlético-mg	73,68%
são paulo	73,68%
corinthians	71,05%
goiás	71,05%
vasco	71,05%
bahia	68,42%
palmeiras	65,79%

Partindo para o nosso segundo classificador, o *Naive Bayes,* os resultados do modelo analisando as métricas que estamos seguindo, *Accuracy* atingiu um valor de 0,6868 com valor para o F1-Score de 0,6932, isso com o mesmo período de treino e validação do classificador anterior.

Observando sua matriz de confusão é possível verificar que o modelo atingiu um total de 261 partidas classificadas corretamente, em um universo de 380 (Partidas de 2019), atingindo assim sua acurácia.

Figura 22 – Matriz de Confusão

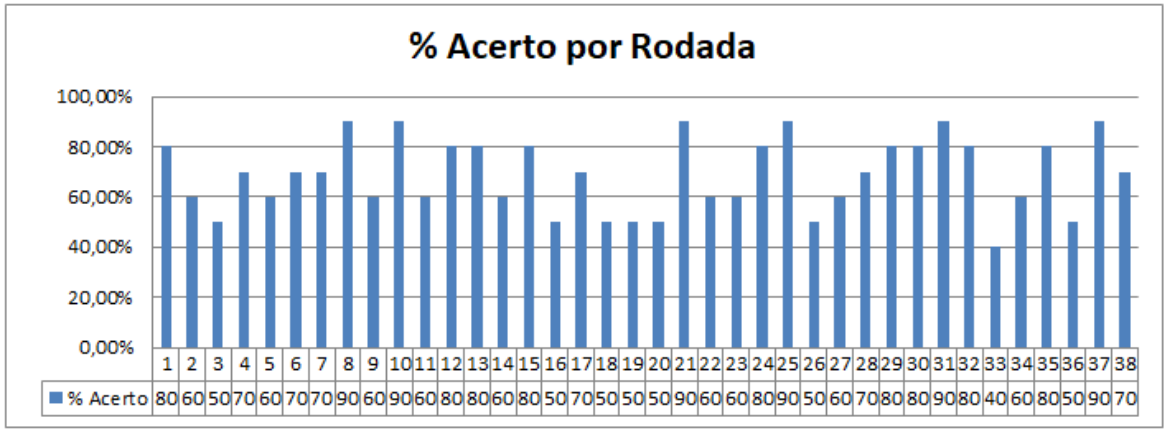
Row ID	M	V	E
M	155	7	22
V	8	58	32
E	33	17	48

Assim como no classificador anterior a maior dificuldade do *Naive Bayes* foi prever as partidas que terminaram empatas, acertando corretamente apenas 48,98% delas, enquanto novamente a classificação dos mandantes vitoriosos teve a

maior parcela de acertos, totalizando 84,24%, a de vitória dos visitantes atingiu 59,18%.

Analisando rodada a rodada, esse classificador não conseguiu prever em 100,00% nenhuma rodada do campeonato, atingindo um valor máximo de acerto em 90,00%, porém, isso ocorreu em outras rodadas, totalizando seis. Seu pior desempenho foi na rodada de número 33, onde atingiu um percentual de acerto de somente 40,00%, ou seja, acertou apenas quatro partidas em dez. Essa rodada contou com seis partidas que terminaram empatadas, talvez seja um dos motivos desse desempenho.

Gráfico 3 - % Acerto por Rodada



Analisando a performance por clube, esse classificador conseguiu melhor desempenho nos dados do time do Botafogo-RJ, com um total de 31 acertos em 38, (81,58%). Já seu pior desempenho foi em três clubes, CSA, Palmeiras e Vasco, com 23 acertos cada (60,53%) conforme a tabela abaixo, com dados de todos os clubes:

Tabela 6 – Lista de acertos por times

Time	ACERTO	% acerto
botafogo-rj	31	81,58%
ceará	30	78,95%
flamengo	29	76,32%
santos	29	76,32%
fortaleza	28	73,68%
athlético-pr	27	71,05%
fluminense	27	71,05%
goiás	27	71,05%
internacional	27	71,05%
avaí	26	68,42%
são paulo	26	68,42%
atlético-mg	25	65,79%
chapecoense	25	65,79%
bahia	24	63,16%
corinthians	24	63,16%
cruzeiro	24	63,16%
grêmio	24	63,16%
csa	23	60,53%
palmeiras	23	60,53%
vasco	23	60,53%

Analisando o terceiro classificador, Redes Neurais Artificiais (PNN), os resultados do modelo apresentaram as seguintes métricas alvos do estudo, *Accuracy* atingiu um valor de 0,6894 com valor para o F1-Score de 0,6922, isso com o mesmo período de treino e validação dos classificadores anteriores, totalizando 262 partidas corretas, conforme matriz de confusão abaixo.

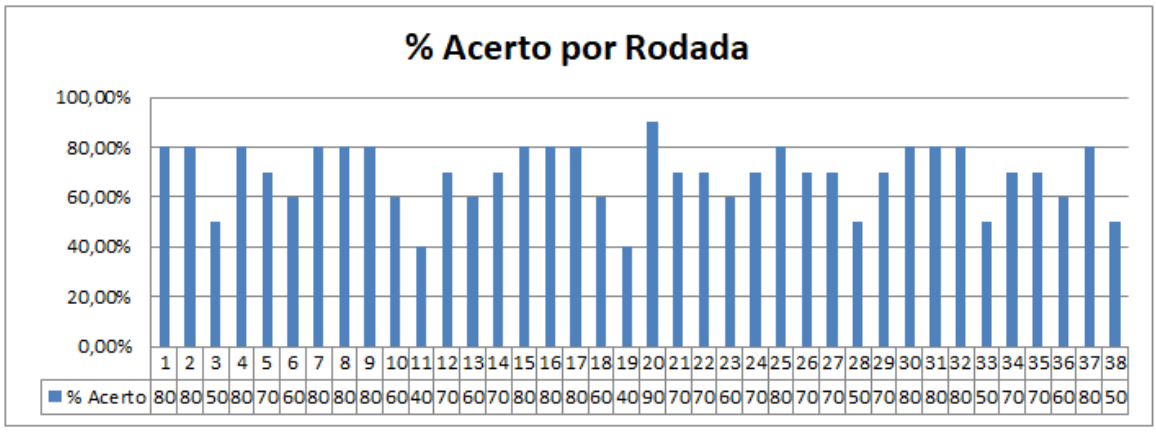
Figura 23 – Matriz de Confusão

Row ID	M	V	E
M	175	3	6
V	16	69	13
E	69	11	18

Esse classificador sofreu bastante para conseguir identificar partidas que terminaram em empate, acertando apenas 18,37% delas, enquanto para verificar as partidas que terminaram com visitante vencedor conseguiu atingir 70,41%. Como os demais até aqui, a maior parcela de acerto foi nos clubes mandantes conseguindo um percentual de acerto de 95,11%.

A rodada onde esse classificador se saiu melhor foi na de número 20, atingindo 90% de desempenho e errando apenas um resultado, esse, porém, era o único empate da rodada e não foi possível acertar. Já seu pior resultado foi de 40% de acerto na rodada 19, onde ocorreu três empates e o modelo não teve êxito na previsão e o outro resultado quando previu empate o mesmo não ocorreu. Abaixo o desempenho do modelo em todas as rodadas.

Gráfico 4 - % Acerto por Rodada



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38
% Acerto	80	80	50	80	70	60	80	80	80	60	40	70	60	70	80	80	80	60	40	90	70	70	60	70	80	70	70	50	70	80	80	80	50	70	70	60	80	50

Verificando o resultado do modelo por time, os melhores foram Botafogo-RJ, CSA e Flamengo, todos com 32 acertos, o que totaliza 84,21% para esses três clubes. Já na ponta inferior da tabela, o time do Corinthians ficou em último com um total de 20 partidas classificadas corretamente chegando apenas a 52,63% de acerto do modelo para esse clube. Abaixo está a lista com todos os demais times.

Tabela 7 – Lista de acertos por times

Time	ACERTO	% acerto
botafogo-rj	32	84,21%
flamengo	32	84,21%
csa	32	84,21%
avaí	31	81,58%
santos	30	78,95%
ceará	27	71,05%
athlético-pr	27	71,05%
goiás	27	71,05%
grêmio	27	71,05%
fortaleza	26	68,42%
fluminense	26	68,42%
internacional	26	68,42%
atlético-mg	26	68,42%
vasco	26	68,42%
são paulo	23	60,53%
bahia	23	60,53%
chapecoense	21	55,26%
cruzeiro	21	55,26%
palmeiras	21	55,26%
corinthians	20	52,63%

Por fim, nosso último classificador, *Gradient Boosted*, que apresentou os seguintes resultados para as métricas alvos do estudo, *Accuracy* de 0,7763 e com valor de F1-Score de 0,7794, isso com o mesmo período de treino e validação dos classificadores anteriores, totalizando 295 partidas corretas, conforme matriz de confusão abaixo.

Figura 24 – Matriz de Confusão

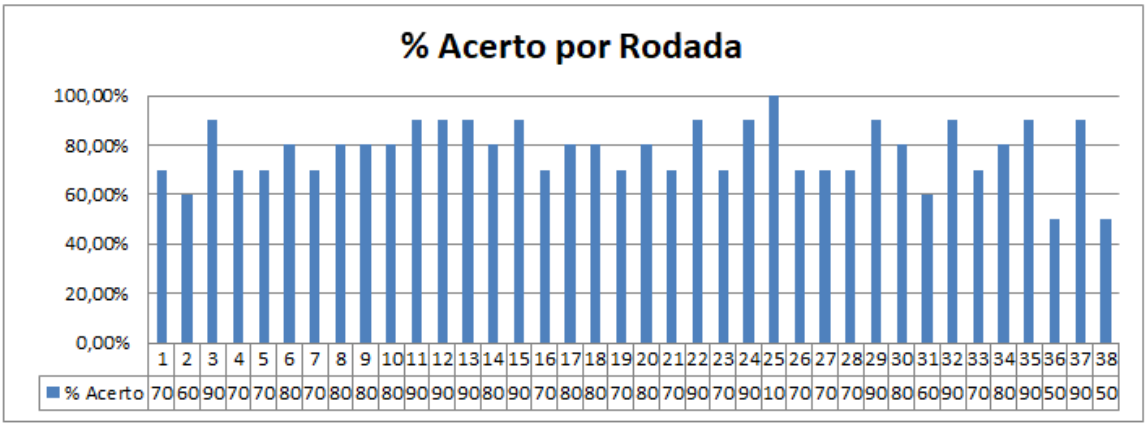
Row ID	M	V	E
M	154	3	27
V	3	67	28
E	18	6	74

Esse classificador, diferentemente dos demais, conseguiu classificar corretamente um número maior de partidas que terminaram empatadas, 74, atingindo 75,51%, marca maior até mesmo das previsões das partidas em que o visitante foi o vencedor (68,37%) e teve um percentual de vitórias do mandante de 83,70%.

Seu melhor desempenho foi na rodada de número 25 onde conseguiu atingir 100,00% de acerto, isso com seis vitórias do mandante, duas do visitante e dois em-

pates. Em mais onze rodadas atingiu a marca de 90,00% de acerto, sendo que suas piores rodadas em termos de classificação correta foram a de número 36 e 38, pois nelas conseguiu um total de 50,00% de acerto, abaixo do desempenho do modelo em todas as rodadas.

Gráfico 5 - % Acerto por Rodada



Analisando por clube, esse modelo classificou corretamente 89,47% dos jogos do Cruzeiro, sendo um total de 34 classificações corretas para um total de 38. Já o pior desempenho foi para a equipe do Atlético-MG, com um total de 22 classificações corretas, 57,89%, abaixo do percentual de acerto de todas as equipes.

Tabela 8 – Lista de acertos por times

Time	ACERTO	% acerto
cruzeiro	34	89,47%
flamengo	33	86,84%
csa	33	86,84%
athlético-pr	33	86,84%
ceará	32	84,21%
são paulo	31	81,58%
grêmio	30	78,95%
botafogo-rj	29	76,32%
avaí	29	76,32%
santos	29	76,32%
goiás	29	76,32%
fortaleza	29	76,32%
fluminense	29	76,32%
internacional	29	76,32%
vasco	29	76,32%
corinthians	29	76,32%
bahia	28	73,68%
chapecoense	28	73,68%
palmeiras	25	65,79%
atlético-mg	22	57,89%

Após a análise dos quatros classificadores propostos no estudo, vamos comparar o desempenho que tiveram com os demais, primeiramente as métricas propostas, *Accuracy* e F1-Score:

Tabela 9– Lista de Classificadores e suas medidas

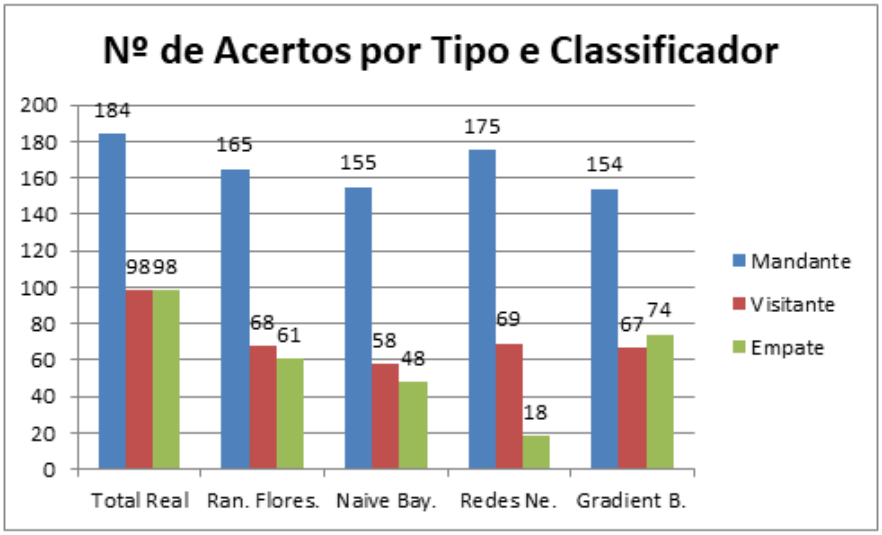
Classificador/Medidas	Acuracia	F1 Score
Randow Florest	0,77368	0,7737
Naive Bayes	0,68684	0,6932
Redes Neurais DDA	0,68947	0,6922
Gradient Boosted	0,77632	0,7794

Como visto na tabela acima, o melhor classificador para o problema proposto foi o *Gradient Boosted*, atingindo uma acurácia de 0,7763. Esse classificador também foi o melhor no F1-Score com 0,7794, notando uma diferença mínima para o *Random Florest*. Em termos de resultado geral, foi 294 de *True Positive* contra 295 do *Gradient Boosted.*

Em comparação com os demais classificadores um diferencial do *Gradient Boosted*, foi conseguir classificar com mais exatidão as partidas que terminaram empatadas com 74 classificações corretas, sendo que o *Random Florest*, segundo nesse quesito, atingiu a marca de 68, conforme gráfico 6.

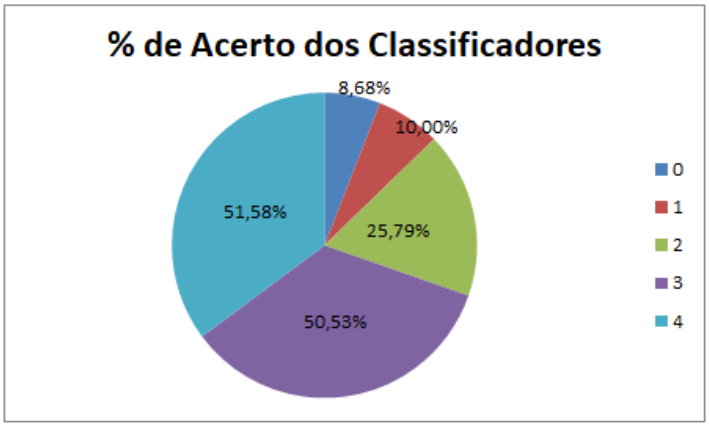
Porém, apesar do ganho na classificação dos jogos que terminaram empatados *Gradient Boosted,* teve o pior desempenho quando o vencedor do jogo foi o mandante, classificando corretamente 154 partidas, contra 175 das Redes Neurais que conforme o gráfico 6, teve melhor desempenho.

Gráfico 6 – Nº de Acertos por tipo e classificador



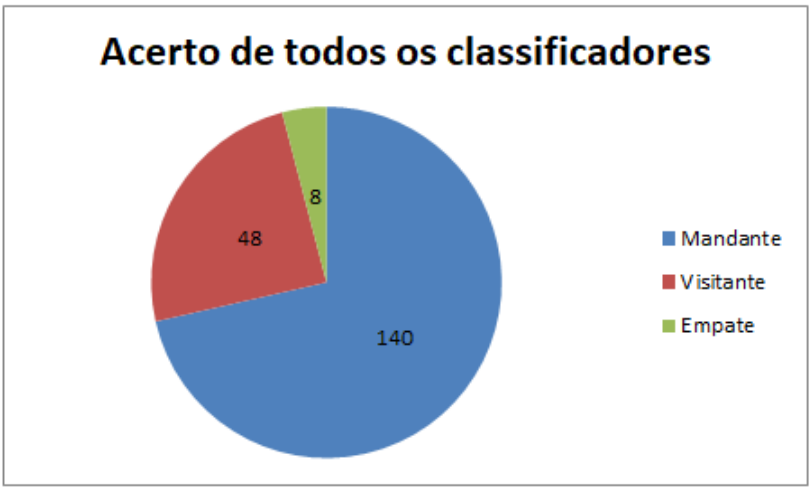
Verificando rodada a rodada o desempenho dos classificadores em apenas 33 jogos ao menos um dos classificadores não acertou o resultado, isso é 8,68%, conforme gráfico abaixo. Por se tratar de futebol, um jogo com muitas variáveis envolvidas onde nem sempre o favorito vence, acreditamos ser um resultado aceitável, pois envolve clubes com jogadores machucados, a serviço das suas seleções ou times em fim de campeonato sem pretenções maiores.

Gráfico 7 – % de Acerto dos classificadores



Em contra ponto, observando as rodadas que todos os classificadores acertaram, chegamos em um total de 196 jogos, 51,57%, ou seja, mais que a metade das partidas. Dessas, somente 8 (4,08%) terminaram em empate e 140 (71,42%) terminaram com a vitória do mandante, conforme gráfico abaixo.

Gráfico 8 – % de Acerto dos classificadores



Olhando para os clubes, o time que teve melhor classificação em seus jogos foi o Flamengo, ficando com 82,24% de acerto em média, enquanto o Palmeiras teve o pior desempenho, 61,84%. Talvez isso se deva pelo fato desse time ter um grande número de empate.

Por fim, o *Gradient Boosted*, teve um melhor desempenho por conseguir classificar corretamente em todos os tipos de vencedores, ficando com o maior número de calssificações corretas (*true positive*) que é o alvo do estudo. É

importante ressaltar que esse trabalho é para fins acadêmicos e que não deve ser levado em conta para apostas esportivas.

Deixo como sugestão para um próximo estudo incluir mais indicadores e variáveis para tentar aumentar o índice de *Accuracy,* haja vista que o futebol possui n variáveis envolvidas, desde a escalação e local do jogo até a previsão do tempo para a data do confronto.

7. Links

Link para o vídeo: youtu.be/hu_eFcVikTE

Link para o repositório: github.com/fernandobrck/tcc_fernando_puc_minas

REFERÊNCIAS

quadrodemedalhas, Disponível em: < http://www.quadrodemedalhas.com/futebol/campeonato-brasileiro/historia-do-campeonato-brasileiro-de-futebol.htm>**.** Acesso em 09 set. 2020.

globoesporte, Disponível em <globoplay.globo.com/v/6669114/>**.** Acesso em 09 set. 2020.

História do Futebol no Brasil. Disponível em: https://rioandlearn.com/pt-br/historia-do-futebol-no-brasil/. Acesso em 13 out 20.