

Projeto Aplicado – Relatório Final

Nome do Aluno	Allan Ferreira
Título do Trabalho	Smart Soccer – Modelo de predição de resultados de futebol com machine learning
Curso	MBA em Machine Learning
Linha de Especialização	Machine Learning Aplicado aos Sistemas de Recomendação
Orientador	Bruno Rafael
Data	20/12/2019

INTRODUÇÃO

1. Apresentação do Desafio e da Solução:

- a. Setor do mercado e a justificativa de tal seleção;
- b. Características e restrições de escopo do Desafio;
- c. Oportunidade vislumbrada que motivou o desenvolvimento da Solução.

-Existe uma prática de negócios chamada trading, que pode ser entendida como os processos de negociação no mercado, tais como compra e venda de ações, títulos, moedas, entre outros. Neste modelo de negócios, o objetivo é a valorização do capital, por isso, é um investimento também, onde se compra um ativo baseado em uma circunstância específica e espera-se que ele se valorize em um determinado período para obter ganho de capital.

-O projeto foi desenvolvido em cima de uma prática que tem crescido muito no Brasil e no mundo, que é o trading esportivo. A diferença para o trading tradicional é que nesta prática os ativos são as probabilidades de evento em um jogo, que variam desde quem vai vencer, número de gols, escanteios, etc. Os usuários tem um cadastro em algum site de bolsa esportiva, onde podem realizar suas apostas antes, durante, ou no início de uma partida.

-A justificativa do projeto se dá pelo grande número de usuários que realizam suas apostas baseadas somente em emoção ou achismo. Vale ressaltar que, mesmo aqueles que tem esta prática como profissão, tem suas técnicas baseadas em achismos e ao acompanhamento em tempo dos jogos e percepção das partidas. Deste modo, o intuito do projeto é auxiliar os usuários com indicadores e recomendações de resultados baseadas em algoritmos de machine learning.

-As características do projeto são auxílios aos apostadores fornecendo recomendação de vencedores de partida de futebol antes do início da partida e no meio da partida.

- A principal restrição é referente ao próprio escopo do cenário, que por se tratar de partidas de futebol, não há como evitar resultados inesperados. Ainda assim, existe um termo muito utilizados pelos usuários do ramo, que se chama “Odd de valor”. Este termo refere-se à um tipo de aposta, que mesmo tendo prejuízo a curto prazo, ela tem valor, e ao decorrer do tempo vai se converter em lucro.

- O sucesso do jogador no trading, é baseado em “know-how”, e as técnicas se limitam à gerência de banca, e planilhas, como foi constatado em vários cursos realizados antes deste trabalho. Por isso, foi realizada uma busca na internet para saber se existiam datasets de partidas de futebol, e constatado que além de existir, eram gratuitos e tinham como

features exatamente as características necessárias para começar a realizar algum tipo de predição e recomendações de resultados.

- Portanto, tendo em vista o acesso aos datasets, e ao cenário atual, onde a maioria dos apostadores realizam suas apostas sem nenhum tipo de fator científico, o produto será um aplicativo, onde o usuário poderá obter recomendações de resultados, com base no histórico do campeonato

2. Identificação da(s) pessoa(s) envolvida(s) no desafio:

- Traders iniciantes, traders profissionais, administradores de sites de recomendação de resultados, entre outras, a seguir temos uma previa de como seria o mapa de empatia do mesmo na tabela 1:

O que eles veem?	O que eles escutam?	O que eles falam?	O que eles fazem?	O que eles pensam?	O que eles sentem?
Falta de informações científicas	Que só começarão a ter lucro depois de muito tempo	É muito difícil "advinhar" resultados	Compram cursos e se frustram	Melhorar a margem de lucro nas apostas realizadas	Medo ao realizar os investimentos na carreira de trader
Falta de ferramentas para o auxílio da tomada de decisão durante a aposta	Que os traders profissionais tem um dom especial	Cada trader tem seu próprio método de predizer	Desistem do mercado, a médio prazo	Diminuir o prejuízo de apostas equivocadas	Medo de que uma hora ou outra perderão toda a banca

Tabela 1-Mapa de empatia

3. Construção da proposta de solução:

- Requisitos da construção do protótipo/MVP, com descrição do experimento e as métricas de validação;
- Indicadores econômico-financeiros do projeto.

- O projeto ainda não está finalizado, visto que ainda é necessário criar as interfaces e infraestrutura de micro-servicos e bancos de dados para criação do aplicativo.

- O primeiro protótipo será um modelo de aprendizado de máquina que dado uma partida de futebol do campeonato espanhol, e dois adversários, retornará, a probabilidade de

número de gols e vencedor da partida, a partir de dados do primeiro tempo. A primeira entrega será através de um aplicativo onde o usuário escolherá uma partida de futebol do dia e o sistema recomendará um vencedor. As métricas para a realização das validações a se utilizar serão a acurácia e a precisão. As escolhas dessas métricas é porque a acurácia, indicará de forma geral o quanto o modelo performou e a precisão pois neste caso, como envolve dinheiro, os falsos positivos são considerados mais prejudiciais que os falsos negativos, ou seja, é melhor deixar passar uma oportunidade boa sem apostar, do que apostar em uma que na verdade é ruim.

- O principal indicador será o retorno, em médio prazo, dos lucros obtidos. As recomendações levarão em conta todos os resultados da temporada anterior.

- Abaixo na tabela 2 seguem estimativas do que será gasto no projeto:

Investimento	Tipo	Valor
Criação de modelo de predição	Desenvolvimento	R\$ 1.000,00
Publicação das APIs	Desenvolvimento	R\$ 500,00
Contratação de domínio	Infra-estrutura	R\$ 60,00
Contratação de servidor + banco de dados	Infra-estrutura	R\$ 450,00
Marketing	Publicidade	R\$ 300,00
	Total:	R\$ 2.310,00

Tabela 2- Indicadores financeiros

Investimento Inicial	R\$ 2.310,00		
Taxa de Desconto	12,00%		
Período (Ano)	Fluxo de Caixa	Valor Presente	VP Acumulado
0	-R\$ 2.310,00	-R\$ 2.310,00	-R\$ 2.310,00
1	R\$ 500,00	R\$ 446,43	-R\$ 1.863,57
2	R\$ 1.000,00	R\$ 797,19	-R\$ 1.066,38
3	R\$ 3.000,00	R\$ 2.135,34	R\$ 1.068,96
4	R\$ 1.500,00	R\$ 953,28	R\$ 2.022,24
5	R\$ 2.000,00	R\$ 1.134,85	R\$ 3.157,09
Soma VPs (Ano 1 a 5)	R\$ 5.467,09		
VPL do Projeto	R\$ 3.157,09		
Taxa Interna de Retorno (TIR)	48,09%		
Taxa de Lucratividade	2,37		
Tempo de Payback	2,50		

Tabela 2.1- Detalhamento retorno financeiro

DESCRIÇÃO DO DESENVOLVIMENTO DO TRABALHO

4. Modelagem arquitetural do Sistema de Recomendação:
 - a. Definição do(s) algoritmos;
 - b. Definição da base de dados;
 - c. Especificação dos resultados esperados e testes iniciais.

No decorrer do desenvolvimento do projeto foram utilizados algoritmos não-supervisionado como o K-Means (este algoritmo funciona agrupando registros de acordo com um conjunto de características, que ele mesmo infere, necessitando apenas ao programador, especificar a quantidade de grupos que se quer extrair), e supervisionados como Regressão Linear (que são algoritmos que tentam prever números de acordo com um conjunto de dados informados já com seus resultados, ou seja, baseado numa série histórica e seus resultados, dado uma entrada nova, tenta prever a saída) e também Naive Bayes (que é um algoritmo de classificação com aprendizagem supervisionada que se baseia no teorema de Bayes, que tem como característica, tentar prever a saída de um conjunto de dados, tendo a premissa de que as características dos conjuntos não tem dependência entre si).

Para o treinamento foram utilizadas bases de dados do site datahub.io, que é um site parceiro oficial de sites governamentais como data.gov.uk e provê também datasets gerais para muitos outros fins, como saúde, educação e também esportes, como futebol. Esta base é muito importante para o projeto pois possui os resultados de todas as partidas de várias temporadas de ligas de futebol, e muitas características que permitem uma grande quantidade de análises de acordo com o objetivo do projeto, ou seja, separadas em antes, meio e fim da partida.

A base de dados utilizada está no formato de CSV e possui 3 datasets de 64 variáveis e 380 registros cada uma, conforme Figura 1 abaixo:

```
8 dsTodos.info(verbose=False)

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 380 entries, 0 to 379
Columns: 61 entries, Div to PSCA
dtypes: float64(36), int64(19), object(6)
memory usage: 181.2+ KB
```

Figura 1 - Visualização das características do dataset

Fonte: Próprio autor

O resultado esperado para o modelo é uma eficiência de 60% de acertos em recomendações que possam auxiliar os traders esportivos nas apostas de partidas de futebol com 3 alternativas para vencedor: **TIME DA CASA**, **VISITANTE** ou **EMPATE**. Durante os testes iniciais obtivemos os seguintes resultados, conforme mostra Figura 2 abaixo:

```
[31] 7 print('previsoes entrando com dados no segundo tempo')
8
9 print("\nAcurácia Gaussian primeiro tempo:",metrics.accuracy_score(y_test_1t, previsoesGaussianNB))
10 print("Recall Gaussian primeiro tempo:",metrics.precision_score(y_test_1t, previsoesGaussianNB,average='macro'))
11 print("Precisão Gaussian primeiro tempo:",metrics.precision_score(y_test_1t, previsoesGaussianNB,average='macro'))
12
13 print("Acurácia SVM primeiro tempo:",metrics.accuracy_score(y_test_1t, previsoesSVM))
14 print("Recall SVM primeiro tempo:",metrics.precision_score(y_test_1t, previsoesSVM,average='macro'))
15 print("Precisão SVM primeiro tempo:",metrics.precision_score(y_test_1t, previsoesSVM,average='macro'))
16
17 print("Acurácia KNeighborsClassifier primeiro tempo:",metrics.accuracy_score(y_test_1t, previsoesKNeighborsClassifier))
18 print("Recall KNeighborsClassifier primeiro tempo:",metrics.precision_score(y_test_1t, previsoesKNeighborsClassifier,average='macro'))
19 print("Precisão KNeighborsClassifier primeiro tempo:",metrics.precision_score(y_test_1t, previsoesKNeighborsClassifier,average='macro'))
20
```

previsoes entrando com dados no segundo tempo

Acurácia Gaussian primeiro tempo: 0.47619047619047616
Recall Gaussian primeiro tempo: 0.31355311355311355
Precisão Gaussian primeiro tempo: 0.31355311355311355
Acurácia SVM primeiro tempo: 0.48412698412698413
Recall SVM primeiro tempo: 0.46666666666666666
Precisão SVM primeiro tempo: 0.3199062233589088
Acurácia KNeighborsClassifier primeiro tempo: 0.44444444444444444
Recall KNeighborsClassifier primeiro tempo: 0.42287878787878785
Precisão KNeighborsClassifier primeiro tempo: 0.4201369436663554

Figura 2- Visualização de acertos com primeiros testes

Fonte: Próprio Autor

Na tabela acima vemos que o algoritmo que teve melhor resultado em termos de acurácia e precisão foi o **SVM**, com 48% de acurácia e 31% de precisão.

Para resolver o problema da baixa acurácia, foi criada uma feature nomeada SUPERTIME. Esta feature, diz com 0 ou 1 se algum dos times é um time muito favorito na liga, como por exemplo, o Barcelona ou Real Madrid é na “La liga”.

A princípio, este critério de supertime foi criado com o ‘know-how’ popular, com um vetor de times, como mostra a Figura 3:

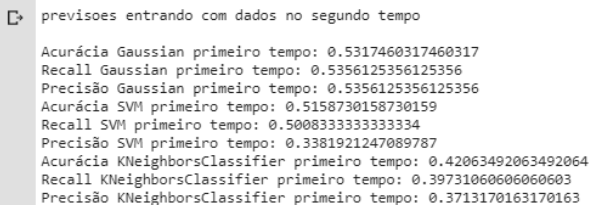
```
[5] 1 def GetGrandeza(time):
2     supertimes= "Paris SG,Lyon,Roma,Milan,Inter,Lazio,Barcelona,Real Madrid,Ath Madrid,Leverkusen,Bayern Munich"
3     timesbons = "FiorentinaCelta de Vigo,Villarreal,Sevilla,Ath Bilbao,Werder Bremen,Wolfsburg,RB Leipzig,Schalke 04"
4     retorno = 3
5     if supertimes.find(time)>-1:
6         retorno = 1
7     if timesbons.find(time)>-1:
8         retorno = 2
9
10    return retorno
11
```

Figura 3 - Definição da função de Super Times

Fonte: Próprio autor

Com esta adição, observamos os resultados abaixo na Figura 4:

```
[36] 2  previsoosSVM = modeloSVM.predict(X_test_1t)
      3  previsoosKNeighborsClassifier = modeloKNeighborsClassifier.predict(X_test_1t)
      4
      5  from sklearn import metrics
      6
      7  print('previsoos entrando com dados no segundo tempo')
      8
      9  print("\nAcurácia Gaussian primeiro tempo:",metrics.accuracy_score(y_test_1t, previsoosGaussianNB))
     10  print("Recall Gaussian primeiro tempo:",metrics.precision_score(y_test_1t, previsoosGaussianNB,average='macro'))
     11  print("Precisão Gaussian primeiro tempo:",metrics.precision_score(y_test_1t, previsoosGaussianNB,average='macro'))
     12
     13  print("Acurácia SVM primeiro tempo:",metrics.accuracy_score(y_test_1t, previsoosSVM))
     14  print("Recall SVM primeiro tempo:",metrics.precision_score(y_test_1t, previsoosSVM,average='macro'))
     15  print("Precisão SVM primeiro tempo:",metrics.precision_score(y_test_1t, previsoosSVM,average='macro'))
     16
     17  print("Acurácia KNeighborsClassifier primeiro tempo:",metrics.accuracy_score(y_test_1t, previsoosKNeighborsClassifier))
     18  print("Recall KNeighborsClassifier primeiro tempo:",metrics.precision_score(y_test_1t, previsoosKNeighborsClassifier,average='macro'))
     19  print("Precisão KNeighborsClassifier primeiro tempo:",metrics.precision_score(y_test_1t, previsoosKNeighborsClassifier,average='macro'))
     20
```



```
previsoos entrando com dados no segundo tempo

Acurácia Gaussian primeiro tempo: 0.5317460317460317
Recall Gaussian primeiro tempo: 0.5356125356125356
Precisão Gaussian primeiro tempo: 0.5356125356125356
Acurácia SVM primeiro tempo: 0.5158730158730159
Recall SVM primeiro tempo: 0.5008333333333333
Precisão SVM primeiro tempo: 0.3381921247089787
Acurácia KNeighborsClassifier primeiro tempo: 0.42063492063492064
Recall KNeighborsClassifier primeiro tempo: 0.39731060606060603
Precisão KNeighborsClassifier primeiro tempo: 0.3713170163170163
```

Figura 4-Resultados com Supertimes(Manual)

Fonte: Próprio autor

A acurácia continuava baixa, então depois do encontro com o orientador, houve a orientação para fazer este calculo de peso de SUPERTIMES automaticamente.

O dataframe foi dividido em 2(visto que tinha que colocar todos na mesma coluna e no dataframe, isto ocorre alternado, pois hora o time está em casa e hora é visitante).

Realizando um agrupamento sobre o montante de VITÓRIAS do time, foi observado o seguinte resultado, conforme Figura 5 :

```
[82] dsTimeHome = dsTodos.copy()
dsTimeHome['NOME'] = dsTimeHome['HomeTeam']

dsTimeHome['VITORIA'] = dsTimeHome.apply(lambda row: [0, 1][row['FTR'] == 0], axis=1)
dsTimeHome['DERROTA'] = dsTimeHome.apply(lambda row: [0, 1][row['FTR'] == 1], axis=1)
dsTimeHome['EMPATE'] = dsTimeHome.apply(lambda row: [0, 1][row['FTR'] == 2], axis=1)

dsTimeFora = dsTodos.copy()
dsTimeFora['NOME'] = dsTimeFora['AwayTeam']

dsTimeFora['VITORIA'] = dsTimeFora.apply(lambda row: [0, 1][row['FTR'] == 1], axis=1)
dsTimeFora['DERROTA'] = dsTimeFora.apply(lambda row: [0, 1][row['FTR'] == 0], axis=1)
dsTimeFora['EMPATE'] = dsTimeFora.apply(lambda row: [0, 1][row['FTR'] == 2], axis=1)

dsTimeUnificado = pd.concat([dsTimeHome, dsTimeFora])

dsTimeUnificado.groupby('NOME')['VITORIA'].sum()
grupo=dsTimeUnificado.groupby('NOME', as_index=False).agg({"VITORIA": "sum"}).sort_values('VITORIA', ascending=False)
```

	NOME	VITORIA
3	Barcelona	82
18	Real Madrid	72
2	Ath Madrid	68
19	Sevilla	55
22	Valencia	50

Figura 5- Unificação de dataset

Fonte:Próprio autor

Um novo dataframe foi criado com os agrupamentos ordenados, de modo que cada time tenha um indice, conforme Figura 6:

```
[16] 1
2 dsTimeUnificado = pd.concat([dsTimeHome, dsTimeFora])
3
4 dsTimeUnificado.groupby('NOME')['VITORIA'].sum()
5 grupo=dsTimeUnificado.groupby('NOME', as_index=False).agg({"VITORIA": "sum"}).sort_values('VITORIA', ascending=False)
6
7 grupo['ID'] = np.arange(len(grupo))
8
9 print(grupo)
```

	NOME	VITORIA	ID
3	Barcelona	26	0
2	Ath Madrid	22	1
13	Real Madrid	21	2
14	Sevilla	17	3
8	Getafe	15	4
16	Valencia	15	5
4	Betis	14	6
7	Espanol	14	7
0	Alaves	13	8
15	Sociedad	13	9
1	Ath Bilbao	13	10

Figura 6-Atribuição de IDs para cada time

Fonte:Próprio autor

Então foi aplicado o algoritmo de clusterização K-Means, que agrupou os times em 7 grupos diferentes, de modo, por exemplo, que o Barcelona ficou isolado.

Com mais esta feature do agrupamento, foi realizado novos testes, com a nova feature, obtendo o seguinte conforme figura 7:

```
[55] 7 print('previsoes entrando com dados no segundo tempo')
8
9 print("\nAcurácia Gaussian primeiro tempo:",metrics.accuracy_score(y_test_1t, previsoesGaussianNB))
10 print("Recall Gaussian primeiro tempo:",metrics.precision_score(y_test_1t, previsoesGaussianNB,average='macro'))
11 print("Precisão Gaussian primeiro tempo:",metrics.precision_score(y_test_1t, previsoesGaussianNB,average='macro'))
12
13 print("Acurácia SVM primeiro tempo:",metrics.accuracy_score(y_test_1t, previsoesSVM))
14 print("Recall SVM primeiro tempo:",metrics.precision_score(y_test_1t, previsoesSVM,average='macro'))
15 print("Precisão SVM primeiro tempo:",metrics.precision_score(y_test_1t, previsoesSVM,average='macro'))
16
17 print("Acurácia KNeighborsClassifier primeiro tempo:",metrics.accuracy_score(y_test_1t, previsoesKNeighborsClassifier))
18 print("Recall KNeighborsClassifier primeiro tempo:",metrics.precision_score(y_test_1t, previsoesKNeighborsClassifier,average='macro'))
19 print("Precisão KNeighborsClassifier primeiro tempo:",metrics.precision_score(y_test_1t, previsoesKNeighborsClassifier,average='macro'))
20
```

```
previsoes entrando com dados no segundo tempo

Acurácia Gaussian primeiro tempo: 0.49206349206349204
Recall Gaussian primeiro tempo: 0.6544578853046595
Precisão Gaussian primeiro tempo: 0.6544578853046595
Acurácia SVM primeiro tempo: 0.5079365079365079
Recall SVM primeiro tempo: 0.49041666666666667
Precisão SVM primeiro tempo: 0.34375
Acurácia KNeighborsClassifier primeiro tempo: 0.42063492063492064
Recall KNeighborsClassifier primeiro tempo: 0.3926515151515151
Precisão KNeighborsClassifier primeiro tempo: 0.3599163679808841
```

Figura 7 – Resultados no intervalo com Super-times(Automático)

Fonte:Próprio autor

```
[62] 4
5 print('\n\nprevisoes inicio partida com base somente histórica')
6
7 print("\nAcurácia Gaussian início partida:",metrics.accuracy_score(y_test_inicio, previsoesGaussianNB))
8 print("Recall Gaussian início partida:",metrics.precision_score(y_test_inicio, previsoesGaussianNB,average='macro'))
9 print("Precisão Gaussian início partida:",metrics.precision_score(y_test_inicio, previsoesGaussianNB,average='macro'))
10
11 print("Acurácia SVM início partida:",metrics.accuracy_score(y_test_inicio, previsoesSVM))
12 print("Recall SVM início partida:",metrics.precision_score(y_test_inicio, previsoesSVM,average='macro'))
13 print("Precisão SVM início partida:",metrics.precision_score(y_test_inicio, previsoesSVM,average='macro'))
14
15 print("Acurácia KNeighborsClassifier início partida:",metrics.accuracy_score(y_test_inicio, previsoesKNeighborsClassifier))
16 print("Recall KNeighborsClassifier início partida:",metrics.precision_score(y_test_inicio, previsoesKNeighborsClassifier,average='macro'))
17 print("Precisão KNeighborsClassifier início partida:",metrics.precision_score(y_test_inicio, previsoesKNeighborsClassifier,average='macro'))
```

```
previsoes inicio partida com base somente histórica

Acurácia Gaussian início partida: 0.3968253968253968
Recall Gaussian início partida: 0.22714097496706195
Precisão Gaussian início partida: 0.22714097496706195
Acurácia SVM início partida: 0.3968253968253968
Recall SVM início partida: 0.3333333333333333
Precisão SVM início partida: 0.13227513227513227
Acurácia KNeighborsClassifier início partida: 0.35714285714285715
Recall KNeighborsClassifier início partida: 0.3261363636363636
Precisão KNeighborsClassifier início partida: 0.29602084922451055
```

Figura 8-Resultados início de partida(Automático)

Fonte:Próprio autor

Na tabela acima vemos que o algoritmo **Naive Bayes** teve melhor desempenho com a classificação automática dos times, com 65% de precisão. Porém , como esperado, por se tratar de uma partida de futebol, como observado, nenhum algoritmo teve bom aproveitando para a predição do resultado com base somente no histórico, no início do

jogo. Então os testes com o início do jogo foram abortados, prosseguindo somente com os testes com entradas no segundo tempo.

5. Definição da principal abordagem técnica a ser utilizada no projeto.
 - a. Seleção do(s) algoritmo(s) que será(ão) utilizado(s) na solução.
 - b. Realização de testes iniciais com a base de dados.

O algoritmo escolhido para ser utilizado na solução final foi o Naive Bayes pois o mesmo foi o que obteve mais precisão.

Portanto foram expandidos os testes para aumentar a base de treinamento e teste. Isto foi feito realizando o download dos datasets da Premier League(campeonato inglês da primeira divisão)

A imagem abaixo demonstra como fora realizado o teste do algoritmo e extraído as métricas selecionadas para a avaliação do modelo, conforme figura 9:

```
[112] 1 previsoesGaussianNB = modeloGaussianNB.predict(X_test_1t)
      2 previsoesSVM = modeloSVM.predict(X_test_1t)
      3 previsoesKNeighborsClassifier = modeloKNeighborsClassifier.predict(X_test_1t)
      4
      5 from sklearn import metrics
      6
      7 print('previsoes entrando com dados no segundo tempo')
      8
      9 print("\nAcurácia Gaussian primeiro tempo:",metrics.accuracy_score(y_test_1t, previsoesGaussianNB))
     10 print("Recall Gaussian primeiro tempo:",metrics.precision_score(y_test_1t, previsoesGaussianNB,average='macro'))
     11 print("Precisão Gaussian primeiro tempo:",metrics.precision_score(y_test_1t, previsoesGaussianNB,average='macro'))
     12
     13 print("Acurácia SVM primeiro tempo:",metrics.accuracy_score(y_test_1t, previsoesSVM))
     14 print("Recall SVM primeiro tempo:",metrics.precision_score(y_test_1t, previsoesSVM,average='macro'))
     15 print("Precisão SVM primeiro tempo:",metrics.precision_score(y_test_1t, previsoesSVM,average='macro'))
     16
     17 print("Acurácia KNeighborsClassifier primeiro tempo:",metrics.accuracy_score(y_test_1t, previsoesKNeighborsClassifier))
     18 print("Recall KNeighborsClassifier primeiro tempo:",metrics.precision_score(y_test_1t, previsoesKNeighborsClassifier,average='macro'))
     19 print("Precisão KNeighborsClassifier primeiro tempo:",metrics.precision_score(y_test_1t, previsoesKNeighborsClassifier,average='macro'))
     20
```

previsoes entrando com dados no segundo tempo

Acurácia Gaussian primeiro tempo: 0.6428571428571429
Recall Gaussian primeiro tempo: 0.5505282331511839
Precisão Gaussian primeiro tempo: 0.5505282331511839

Figura 9- Testes com campeonato inglês

Fonte:Próprio autor

6. Implementação da solução utilizando algum framework de algoritmos de Machine Learning.
 - a. Configuração do ambiente de desenvolvimento.
 - b. Implementação ou seleção do(s) algoritmo(s).
 - c. Validação e teste do algoritmo com dados reais.

Para o desenvolvimento do modelo foi utilizada ferramenta Google Colab que é um ambiente de notebooks Jupyter que não requer configuração e é executado na nuvem, encontrado na url <https://colab.research.google.com>:

- Para a codificação do modelo foi utilizado as bibliotecas do pandas para a utilização do dataset e scikit-learn para os algoritmos a serem avaliados, a imagem abaixo descreve como foi implementado as chamadas aos algoritmos, realizados os testes e validações.

▼ Amostragem

Home=0 Away=1 Draw=2

```
[113] 1 #supertime
2 treinoDataAtePrimeiroTempo = pd.DataFrame(dsTratado, columns = ['HTHG', 'HTAG', 'HS', 'AS', 'GH', 'GA'])
3
4 #sem supertime
5 #treinoDataAtePrimeiroTempo = pd.DataFrame(dsTratado, columns = ['HTHG', 'HTAG', 'HS', 'AS'])
6
7
8 treinoTargetAtePrimeiroTempo = pd.DataFrame(dsTratado, columns = ['VENC'])
9
10 from sklearn.model_selection import train_test_split
11 X_train_it, X_test_it, y_train_it, y_test_it = train_test_split(treinoDataAtePrimeiroTempo, \
12                                                                  treinoTargetAtePrimeiroTempo, test_size=0.33, random_state=44)
13
14
15 treinoDataInicioPartida = pd.DataFrame(dsTratado, columns = ['GH', 'GA'])
16 treinoTargetInicioPartida = pd.DataFrame(dsTratado, columns = ['VENC'])
17
18 from sklearn.model_selection import train_test_split
19 X_train_inicio, X_test_inicio, y_train_inicio, y_test_inicio = train_test_split(treinoDataInicioPartida, \
20                                                                                  treinoTargetInicioPartida, test_size=0.33, random_state=44)
```

Figura 10-Amostragem dos dados para treinamento

Fonte: Próprio autor

Conforme figura 10, os dados amostrados estão considerando o fator de super-times, quantidade de gols no primeiro tempo, chutes a gol e como resultado, o fator 'VENC', que se refere ao time que saiu vitorioso ao final da partida.

Já a figura 11 abaixo, mostra como foram treinados os modelos para predição tanto dos resultados no intervalo, quanto no início da partida(que foram abortados pela baixa quantidade de acertos)

Treinamento

```
[114] 1 from sklearn.naive_bayes import GaussianNB
2 modeloGaussianNB = GaussianNB()
3 modeloGaussianNB.fit(X_train_1t,y_train_1t)
4
5 from sklearn import svm
6 modeloSVM = svm.SVC(kernel='linear', C = 1.0)
7 modeloSVM.fit(X_train_1t,y_train_1t)
8
9 from sklearn.neighbors import KNeighborsClassifier
10 modeloKNeighborsClassifier = KNeighborsClassifier(n_neighbors=3)
11 modeloKNeighborsClassifier.fit(X_train_1t,y_train_1t)
12
13
14 from sklearn.naive_bayes import GaussianNB
15 modeloGaussianNB_inicio = GaussianNB()
16 modeloGaussianNB_inicio.fit(X_train_inicio,y_train_inicio)
17
18 from sklearn import svm
19 modeloSVM_inicio = svm.SVC(kernel='linear', C = 1.0)
20 modeloSVM_inicio.fit(X_train_inicio,y_train_inicio)
21
22 from sklearn.neighbors import KNeighborsClassifier
23 modeloKNeighborsClassifier_inicio = KNeighborsClassifier(n_neighbors=3)
24 modeloKNeighborsClassifier_inicio.fit(X_train_inicio,y_train_inicio)
```

Figura 11-Treinamento do modelo

Fonte:Próprio autor

A figura 12 abaixo mostra como foram realizadas as métricas dos algoritmos selecionados:

▼ Métricas

```
[115] 1 previsoesGaussianNB = modeloGaussianNB.predict(X_test_1t)
2 previsoesSVM = modeloSVM.predict(X_test_1t)
3 previsoesKNeighborsClassifier = modeloKNeighborsClassifier.predict(X_test_1t)
4
5 from sklearn import metrics
6
7 print('previsoes entrando com dados no segundo tempo')
8
9 print("\nAcurácia Gaussian primeiro tempo:",metrics.accuracy_score(y_test_1t, previsoesGaussianNB))
10 print("Recall Gaussian primeiro tempo:",metrics.precision_score(y_test_1t, previsoesGaussianNB,average='macro'))
11 print("Precisão Gaussian primeiro tempo:",metrics.precision_score(y_test_1t, previsoesGaussianNB,average='macro'))
12
13 print("Acurácia SVM primeiro tempo:",metrics.accuracy_score(y_test_1t, previsoesSVM))
14 print("Recall SVM primeiro tempo:",metrics.precision_score(y_test_1t, previsoesSVM,average='macro'))
15 print("Precisão SVM primeiro tempo:",metrics.precision_score(y_test_1t, previsoesSVM,average='macro'))
16
17 print("Acurácia KNeighborsClassifier primeiro tempo:",metrics.accuracy_score(y_test_1t, previsoesKNeighborsClassifier))
18 print("Recall KNeighborsClassifier primeiro tempo:",metrics.precision_score(y_test_1t, previsoesKNeighborsClassifier,average='macro'))
19 print("Precisão KNeighborsClassifier primeiro tempo:",metrics.precision_score(y_test_1t, previsoesKNeighborsClassifier,average='macro'))
20
```

Figura 12- Avaliação de métricas para o modelo

Fonte:Próprio autor

A figura 13 mostra como foram realizados os testes, o usuário informando o time da casa, o time visitante, quantidade de gols e chutes realizados no primeiro tempo, e com base nestas informações o sistema recomenda HOME(Time da casa vence), AWAY(Time visitante vence) ou DRAW(Empate)

▼ Testando

```
[139] 1 modelo = modeloGaussianNB
      2
      3 timeHome='Barcelona'
      4 timeAway='Real Madrid'
      5
      6 chutesAGolHome=4
      7 chutesAGolAway=2
      8
      9 golsHome=0
     10 golsAway=0
     11
     12 jogoAtual = [[golsHome,\
     13               golsAway,\
     14               chutesAGolHome,\
     15               chutesAGolAway,\
     16               GetGrandezaCluster(timeHome),\
     17               GetGrandezaCluster(timeAway)]]
     18 previsoos = modelo.predict(jogoAtual)
     19
     20 print('A previsão é',DesConverter(previsoos))
```

☞ A previsão é Home

Figura 13- Teste do modelo

Fonte:Próprio autor

7. Avaliação da solução proposta.

- Definição da medida de acurácia a ser aplicada sobre os resultados gerados pelo(s) algoritmo(s).
- Realização de experimentos e coleta dos resultados.
- Ajuste de parâmetros.

A técnica para a avaliação dos resultados foi a acurácia, precisão e recall. A acurácia irá responder o quão frequente o modelo está, já a precisão responde das partidas, quantas realmente tiveram o resultado acertado. O recall por sua vez, mostra dentre todas as situações de classe positivo como valor esperado, quantas estão corretas. Tendo essas perguntas a tabela abaixo demonstra os resultados individuais para cada algoritmo

Algoritmo	Sem feature supertime			Com feature supertime		
	Acurácia	Precisão	Recall	Acurácia	Precisão	Recall
Naive Bayes	0.47	0.31	0.31	0.49	0.64	0.65
SVM	0.48	0.31	0.46	0.50	0.34	0.49
KNeighborsClassifier	0.44	0.42	0.42	0.42	0.35	0.39

Tabela 3-Comparação de performance dos algoritmos

Fonte: Próprio autor

Na documentação d Scikit-learn. o GaussianNB, não tem nenhum hyperparâmetro para ajuste, então não foi possível realizar o tuning com o GaussianNB. Porém para questões didáticas, foi realizado com o algoritmo SVM, como segue abaixo na imagem,e constatando que ainda assim, a melhor precisão continuava sendo do Naive Bayes, como observa-se na figura 14:

```
from sklearn.model_selection import GridSearchCV |

param_grid = {'C': [0.1, 1, 10, 100, 1000],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'kernel': ['rbf']}

grid = GridSearchCV(svm.SVC(), param_grid, refit = True, verbose = 3)

grid.fit(X_train_it, y_train_it)

[308] print(grid.best_params_)

print(grid.best_estimator_)

grid_predictions = grid.predict(X_test_it)

print(classification_report(y_test_it, grid_predictions))

{ 'C': 1000, 'gamma': 0.0001, 'kernel': 'rbf' }
SVC(C=1000, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.0001, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
      precision    recall  f1-score   support

         0         0.49      0.94      0.64         50
         1         0.53      0.50      0.52         32
         2         0.00      0.00      0.00         44

 accuracy          0.50         126
 macro avg         0.34         0.48         0.39         126
 weighted avg         0.33         0.50         0.39         126
```

Figura 14- Realização de ajuste de hiperparâmetros com SVM

Fonte: Próprio autor

DESCRIÇÃO DO TRABALHO FINAL

8. Fase de Preparação:

- Apresentação das ferramentas;
- Apresentação das técnicas;
- Apresentação das integrações;
- Especificação de algoritmos e frameworks utilizados;
- Apresentação das fontes de dados.

Durante a fase de preparação dos dados foram utilizados os recursos disponíveis na biblioteca do pandas que é uma biblioteca open source voltada para análise de dados que torna fácil a manipulação de conjuntos de dados, abaixo na figura 15 pode-se observar algumas utilizações da mesma com algumas variáveis do conjunto de dados.

```
[31] 1
2 dslaliga1617 = pd.read_csv('laliga-1617_csv.csv')
3 dslaliga1718 = pd.read_csv('laliga-1718_csv.csv')
4 dslaliga1819 = pd.read_csv('laliga-1819_csv.csv')
5
6 dsPremier1617 = pd.read_csv('season-1617_csv.csv')
7 dsPremier1718 = pd.read_csv('season-1718_csv.csv')
8 dsPremier1819 = pd.read_csv('season-1819_csv.csv')
9
10 #dsTodos = pd.concat([ dslaliga1819,dslaliga1617,dslaliga1718,dsPremier1617,dsPremier1718,dsPremier1819])
11 dsTodos = pd.concat([ dslaliga1819,dslaliga1617,dslaliga1718])
12 #dsTodos=dsLaliga1819.copy()
13 dsTodos.info(verbose=False)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1140 entries, 0 to 379
Columns: 64 entries, AC to WHH
dtypes: float64(39), int64(19), object(6)
memory usage: 578.9+ KB
```

```
[30] 1 dsTodos.sample(5)
```

	Div	Date	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	HS	AS	HST	AST	HF	AF	HC	AC	HY	AY	HR	AR	B365H	B36
248	SP1	24/02/2019	Valladolid	Betis	0	2	A	0	1	A	10	6	2	3	10	9	13	4	1	2	0	0	2.70	:
27	SP1	02/09/2018	Betis	Sevilla	1	0	H	0	0	D	11	8	3	2	15	17	5	8	3	6	0	1	2.80	:
1	SP1	17/08/2018	Girona	Valladolid	0	0	D	0	0	D	13	2	1	1	21	20	3	2	1	1	0	0	1.75	:
14	SP1	25/08/2018	Valladolid	Barcelona	0	1	A	0	0	D	7	10	4	5	13	11	6	10	1	1	0	0	17.00	:
224	SD1	09/09/2019	Girona	Huesca	0	2	A	0	2	A	13	6	7	6	0	18	13	2	3	1	1	0	2.00	:

Figura 15-Importação de visualização do dataset

Fonte:Próprio autor

A imagem 15 acima mostra que o dataset completo possui 11 variáveis e 1140 registros, referente aos dados das partidas do campeonato espanhol.

A imagem 16 abaixo mostra um conjunto de 10 registro aleatórios contidos no dataset, já depois de tratado e com as variáveis de supertime:

```
[32] 1 dsTratado.sample(10)
```

	HomeTeam	AwayTeam	FTHG	FTAG	HTHG	HTAG	HS	HTR	AS	B365H	B365A	B365D	FTR	GH	GA
120	Leganes	Alaves	1	0	1	0	13	H	7	2.20	3.60	3.1	H	2	4
299	Sociedad	Betis	2	1	1	0	15	H	11	2.25	3.30	3.3	H	4	7
75	Espanol	Villarreal	3	1	1	1	28	D	13	2.25	3.40	3.2	H	7	6
227	Sevilla	Eibar	2	2	0	1	13	A	8	1.66	5.00	4.0	D	1	2
229	Alaves	Levante	2	0	1	0	16	H	8	2.00	3.80	3.4	H	4	2
193	Real Madrid	Sevilla	2	0	0	0	23	D	4	1.70	4.50	4.2	H	5	1
318	Valladolid	Getafe	2	2	1	1	8	D	8	3.20	2.55	2.9	D	6	1
279	Villarreal	Vallecano	3	1	0	1	12	A	7	1.60	5.50	4.2	H	6	0
307	Getafe	Ath Bilbao	1	0	0	0	6	D	9	2.30	3.50	3.0	H	1	4
163	Ath Madrid	Espanol	1	0	0	0	8	D	18	1.40	10.00	4.0	H	5	7

Figura 16-Amostra de dados do dataset tratado

Fonte: Próprio autor

9. Fase de Processamento.

- Apresentação das ferramentas;
- Apresentação das técnicas;
- Apresentação das integrações;
- Especificação de algoritmos e frameworks utilizados;
- Apresentação das fontes de dados.

Para a fase de processamento o conjunto de dados foi dividido em 70% para treinamento e 30% para testes, conforme mostra figura 17 abaixo, separando em dados para treino com informações do segundo tempo e outro para início de partida baseado em histórico.

Amostragem

Home=0 Away= 1 Draw=2

```
[69] 1 #supertime
2 treinoDataAtePrimeiroTempo = pd.DataFrame(dsTratado, columns = ['HTHG', 'HTAG', 'HS', 'AS', 'GH', 'GA'])
3
4 #sem supertime
5 treinoDataAtePrimeiroTempo = pd.DataFrame(dsTratado, columns = ['HTHG', 'HTAG', 'HS', 'AS'])
6
7
8 treinoTargetAtePrimeiroTempo = pd.DataFrame(dsTratado, columns = ['VENC'])
9
10 from sklearn.model_selection import train_test_split
11 X_train_it, X_test_it, y_train_it, y_test_it = train_test_split(treinoDataAtePrimeiroTempo, \
12 | treinoTargetAtePrimeiroTempo, test_size=0.30, random_state=44)
13
14
15 treinoDataInicioPartida = pd.DataFrame(dsTratado, columns = ['GH', 'GA'])
16 treinoTargetInicioPartida = pd.DataFrame(dsTratado, columns = ['VENC'])
17
18 from sklearn.model_selection import train_test_split
19 X_train_inicio, X_test_inicio, y_train_inicio, y_test_inicio = train_test_split(treinoDataInicioPartida, \
20 | treinoTargetInicioPartida, test_size=0.30, random_state=44)
```

Figura 17-Processo de amostragem dos dados

Fonte:Próprio autor

Após essa divisão de treino e teste foram implementados os algoritmos para o processamento e realizado o treinamento do mesmo, conforme seguem imagens abaixo:

▼ Funções

```
[36] 1 def GetGrandeza(time):
2     supertimes= "Paris SG,Lyon,Roma,Milan,Inter,Lazio,Barcelona,Real Madrid,Ath Madrid,Leverkusen,Bayern Munich"
3     timesbons = "FiorentinaCelta de Vigo,Villarreal,Sevilla,Ath Bilbao,Werder Bremen,Wolfsburg,RB Leipzig,Schalke 04"
4     retorno = 3
5     if supertimes.find(time)>-1:
6         retorno = 1
7     if timesbons.find(time)>-1:
8         retorno = 2
9
10    return retorno
11
12 def Converter(text):
13     retorno = '-1'
14     if text == 'A':
15         retorno=1
16     elif text == 'D':
17         retorno = 2
18     elif text == 'H':
19         retorno = 0
20     return retorno
21
```

Figura 18-Funções para tratamento dos dados

Fonte:Próprio autor

Na figura 18 acima, são apresentadas as funções GetGrandeza(),que se refere a primeira função utilizada para obter a feature de supertime, e também a função Converter(), que serve para transformar em atributo numérico a feature que vem no dataset como vencedor da partida, pois vem como alfanumérico(A,D,H).

```
38 def DesConverter(text):
39     retorno = '-1'
40     if text == 1:
41         retorno='Away'
42     elif text == 2:
43         retorno = 'Draw'
44     elif text == 0:
45         retorno = 'Home'
46     return retorno
47
48 def GetGrandezaCluster(time):
49     grupo['cluster'] = kmeans.labels_
50     retorno = grupo[grupo['NOME']==time]['cluster']
51     return int(retorno)
```

Figura 19-Funções para tratamento dos dados

Fonte:Próprio autor

Na figura 19 acima, são apresentadas as funções DesConverter(), que é utilizada para mostrar ao usuário a recomendação do resultado da partida, visto que os algoritmos

processam com atributos numéricos, essa conversão se fez necessária. Já a função GetGrandeza(), atribui a feature cluster, o valor de grandeza de supertime, já processado e tratado como pode-se observar na figura 20 a seguir o processo:

```
[47] 1 dsTimeHome = dsTratado.copy()
2 dsTimeHome['NOME'] = dsTimeHome['HomeTeam']
3
4 dsTimeHome['VITORIA'] = dsTimeHome.apply(lambda row: [0, 1][row['VENC'] == 0], axis=1)
5 dsTimeHome['DERROTA'] = dsTimeHome.apply(lambda row: [0, 1][row['VENC'] == 1], axis=1)
6 dsTimeHome['EMPATE'] = dsTimeHome.apply(lambda row: [0, 1][row['VENC'] == 2], axis=1)
7
8 dsTimeFora = dsTratado.copy()
9 dsTimeFora['NOME'] = dsTimeFora['AwayTeam']
10
11 dsTimeFora['VITORIA'] = dsTimeFora.apply(lambda row: [0, 1][row['VENC'] == 1], axis=1)
12 dsTimeFora['DERROTA'] = dsTimeFora.apply(lambda row: [0, 1][row['VENC'] == 0], axis=1)
13 dsTimeFora['EMPATE'] = dsTimeFora.apply(lambda row: [0, 1][row['VENC'] == 2], axis=1)
14
15 dsTimeHome
```

Figura 20-Transformando dataset Casa e Visitante em um só

Fonte:Próprio autor

Na figura 20 acima, o dataset é dividido em duas partes, tratando as colunas (VITORIA,DERROTA,EMPATE) para ter o mesmo valor, independente de o time estar em casa ou fora.Isto porque no dataset original, há duas colunas para os nomes dos times chamadas HomeTeam e AwayTeam, e o objetivo aqui era ter uma coluna somente por time.

```
[48] 1
2 dsTimeUnificado = pd.concat([dsTimeHome, dsTimeFora])
3
4 dsTimeUnificado.groupby('NOME')['VITORIA'].sum()
5 grupo=dsTimeUnificado.groupby('NOME', as_index=False).agg({"VITORIA": "sum"}).sort_values('VITORIA',ascending=False)
6
7 grupo['ID'] = np.arange(len(grupo))
8
9 print(grupo)
```

	NOME	VITORIA	ID
3	Barcelona	26	0
2	Ath Madrid	22	1
13	Real Madrid	21	2
14	Sevilla	17	3
8	Getafe	15	4
16	Valencia	15	5
4	Betis	14	6
7	Espanol	14	7
0	Alaves	13	8
15	Sociedad	13	9
1	Ath Bilbao	13	10
6	Eibar	11	11
11	Leganes	11	12

Figura 21- Unificação de dataset e atribuição de Ids

Fonte:Próprio autor

Na figura acima, pode se observar a concatenação dos datasets para unificação, e a criação da nova coluna VITORIA, que serve para agrupar e armazenar a quantidade de vitórias que o time teve no dataset(independente se em casa ou fora). Na linha 7, utilizou-se a biblioteca NUMPY para fazer a atribuição sequencial dos Ids para cada time.

10. Fase de Armazenamento.

- Apresentação das ferramentas;
- Apresentação das técnicas;
- Apresentação das integrações;
- Especificação de algoritmos e frameworks utilizados;
- Apresentação das fontes de dados.

Os dados foram obtidos através de datasets em formato CSV no site datahub.io (que é um site parceiro oficial de sites governamentais como data.gov.uk e provê também datasets gerais para muitos outros fins, como saúde, educação e também esportes, como futebol). Os arquivos estão no formato de CSV e possuem 64 variáveis e 380 registros cada um. Para a utilização no projeto, o escopo das variáveis, no tratamento inicial foi reduzido à 8 features, como mostra figura 22 abaixo:

```
[73] 1 dsTratado = dsTodos[['HomeTeam', 'AwayTeam', 'FTHG', 'FTAG', 'HTHG', 'HTAG', 'HS', 'HTR', 'AS', 'FTR']].copy()
      2 dsTratado
```

	HomeTeam	AwayTeam	FTHG	FTAG	HTHG	HTAG	HS	HTR	AS	FTR
0	Betis	Levante	0	3	0	1	22	A	6	A
1	Girona	Valladolid	0	0	0	0	13	D	2	D
2	Barcelona	Alaves	3	0	0	0	25	D	3	H
3	Celta	Espanol	1	1	0	1	12	A	14	D
4	Villarreal	Sociedad	1	2	1	1	16	D	8	A
...
375	Levante	Ath Madrid	2	2	2	0	17	H	17	D
376	Sevilla	Ath Bilbao	2	0	1	0	9	H	9	H
377	Valladolid	Valencia	0	2	0	1	19	A	9	A
378	Eibar	Barcelona	2	2	2	2	15	D	6	D
379	Real Madrid	Betis	0	2	0	0	9	D	9	A

380 rows x 10 columns

Figura 22-Escopo dos datasets

Fonte:Próprio autor

Atualmente todos os dados estão sendo salvos em arquivos CSV's.

Para futura integrações e criação o modelo poderá ser disponibilizado uma API para a realização de integrações e recomendações em tempo real.

11. Fase de Visualização dos Dados.

- Apresentação das ferramentas;
- Apresentação das técnicas;
- Apresentação das integrações;
- Especificação de algoritmos e frameworks utilizados;

Foram trabalhados novas variáveis para o auxílio do algoritmo, conforme mostra explicação abaixo na figura 23:

No código abaixo:

Adicionada coluna VENC pra expressar de forma numerica o ganhador da partida

Atributos GH,e GA (Grandeza Home e Grandeza Away) é calculada de acordo com o 'feeling'

coluna VENC_1T para expressar em forma numerica quem esta ganhando no primeiro tempo

campos BOLSACASA, EMPATE e FORA foram pra expressar a ODD inicial da casa de apostas

```
[45] 1 dsTratado['GH'] = dsTratado.apply(lambda row: GetGrandeza(row.HomeTeam), axis = 1)
2 dsTratado['GA'] = dsTratado.apply(lambda row: GetGrandeza(row.AwayTeam), axis = 1)
3 dsTratado['VENC'] = dsTratado.apply(lambda row: Converter(row.FTR), axis = 1)
4 dsTratado['VENC_1T'] = dsTratado.apply(lambda row: Converter(row.HTR), axis = 1)
5 dsTratado['BOLSACASA'] = dsTratado.apply(lambda row: [0, 1][min(row.B365H,row.B365A,row.B365D)==row.B365H], axis=1)
6 dsTratado['BOLSAFORA'] = dsTratado.apply(lambda row: [0, 1][min(row.B365H,row.B365A,row.B365D)==row.B365A], axis=1)
7 dsTratado['BOLSAEMPATE'] = dsTratado.apply(lambda row: [0, 1][min(row.B365H,row.B365A,row.B365D)==row.B365D], axis=1)
8
9 dsTratado
```

Figura 23-Limpeza e tratamento dos dados

Fonte:Próprio autor

Foram utilizadas as bibliotecas matplotlib(que é uma biblioteca para a visualização de dados em Python e apresenta uma API orientada a objetos que permite a criação de gráficos em 2D de uma forma simples,de diversos tipos de gráficos, como em barra, em linha, em pizza, histogramas entre outras opções) e seaborn(que é uma biblioteca que atua em cima do matplotlib e ajuda a melhorar o visual dos gráficos, dando uma aparência mais bem acabada) para a geração de gráficos de cada uma das variáveis, realização de amostragens das mesmas explicando sua importância ou não dentro do conjunto de dados.

Na figura 24 abaixo, utilizando as 2 bibliotecas, pode se observar o agrupamento da variável de supertime(cluster) de acordo com o número de vitórias do dataset.


```
dfCluster['cluster'] = kmeans.labels_
sb.pairplot(dfCluster, hue='cluster')
```

```

/usr/local/lib/python3.6/dist-packages/numpy/core/_methods.py:217: RuntimeWarning: Degrees of freedom
keepdims=keepdims)
/usr/local/lib/python3.6/dist-packages/numpy/core/_methods.py:209: RuntimeWarning: invalid value encour
ret = ret.dtype.type(ret / rcount)
/usr/local/lib/python3.6/dist-packages/statsmodels/nonparametric/kde.py:487: RuntimeWarning: invalid va
binned = fast_linbin(X, a, b, gridsize) / (delta * nobs)
/usr/local/lib/python3.6/dist-packages/statsmodels/nonparametric/kdetools.py:34: RuntimeWarning: inval
FAC1 = 2*(np.pi*bw/RANGE)**2
<seaborn.axisgrid.PairGrid at 0x7f30a8c797b8>

```

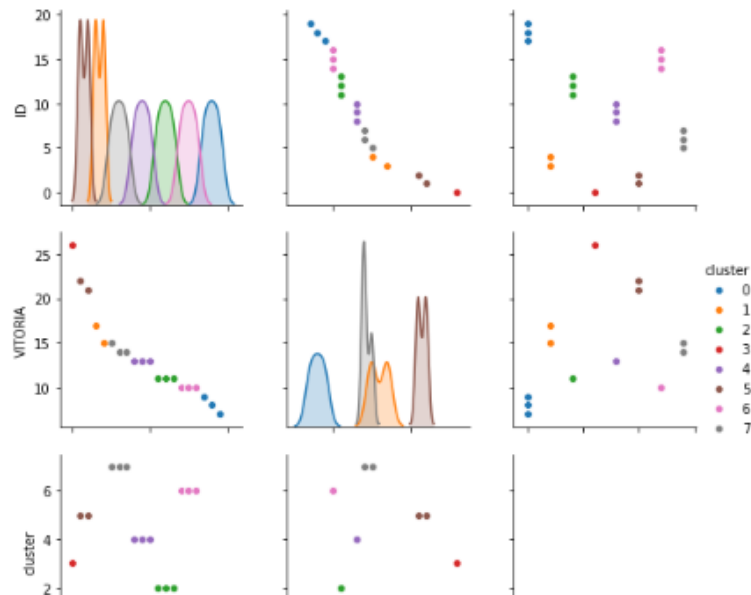


Figura 24- Gráfico de incidência de vitórias x grupo de supertime

Fonte: Próprio autor

```

%matplotlib inline
# passando os valores de x e y como Dataframes
dfRegressaoPlot = grupo
X = dfRegressaoPlot[['ID']]
Y = dfRegressaoPlot[['VITORIA']]
# criando e treinando o modelo
model = LinearRegression()
model.fit(X, Y)
Y_pred = model.predict(X)
pl.scatter(X, Y)

pl.plot(X, Y_pred, color='red')
pl.show()

```

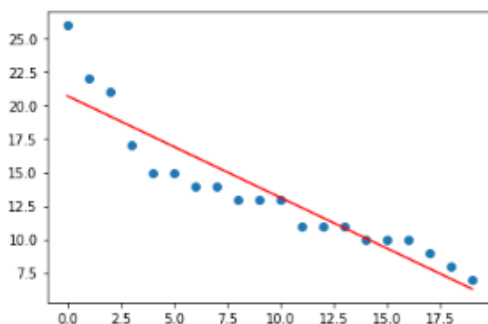


Figura 25- Plotagem de Regressão Linear com agrupamento de supertimes

Fonte: Próprio autor

Pode se observar pela figura 25, que há uma grande diferença entre os três primeiros times do campeonato, e o restante. Esta é basicamente a especificação automática de Supertimes que o trabalho buscava nos testes iniciais.

```
import pandas_profiling as pp
pp.ProfileReport(dsTratado)
```

Figura 26- Realização de Profiling

Fonte: Próprio autor

A imagem acima mostra a realização de um “Profiling”, que é um recurso dos Pandas que mostra vários aspectos da base de dados para análise, conforme seguem abaixo:

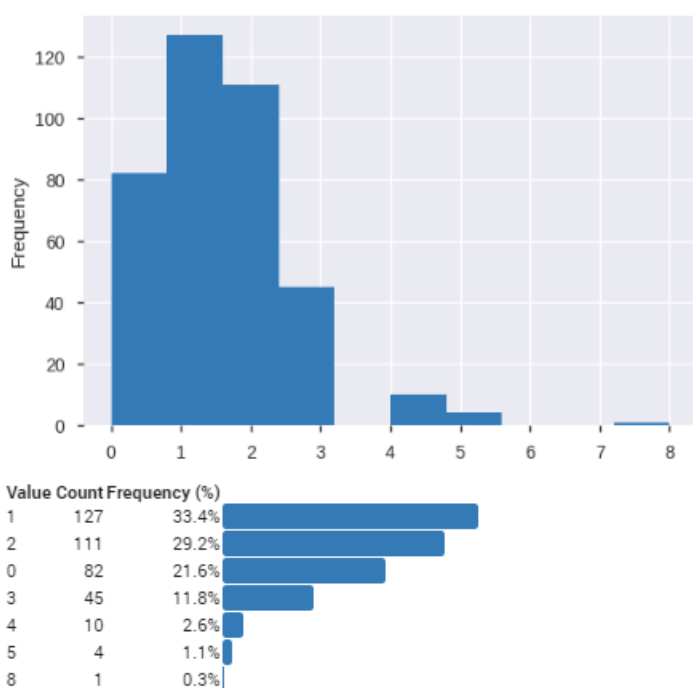


Figura 27-Análise da coluna FTG(Quantidade final de gols)

Fonte: Próprio autor

Na figura 27 acima, foi analisado o campo FTG, e percebe-se que 33% das partidas tem somente 1 gol durante todo o decorrer.

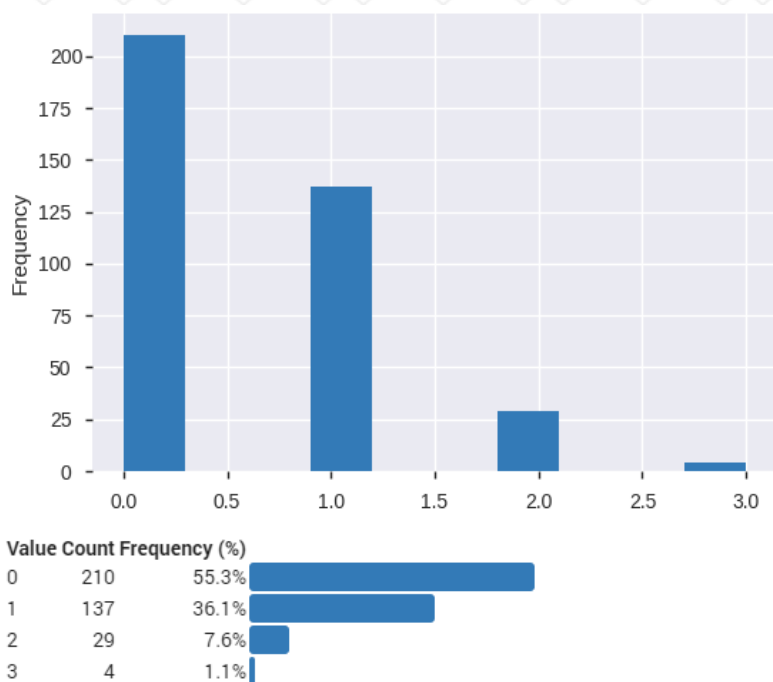


Figura 28- Análise da feature HTHG(Número de gols do time da casa no primeiro tempo)

Fonte: Próprio autor

A figura 28 acima analise o campo HTHG,e revela que em mais da metade dos jogos, o time da casa vira o primeiro tempo sem fazer gols.

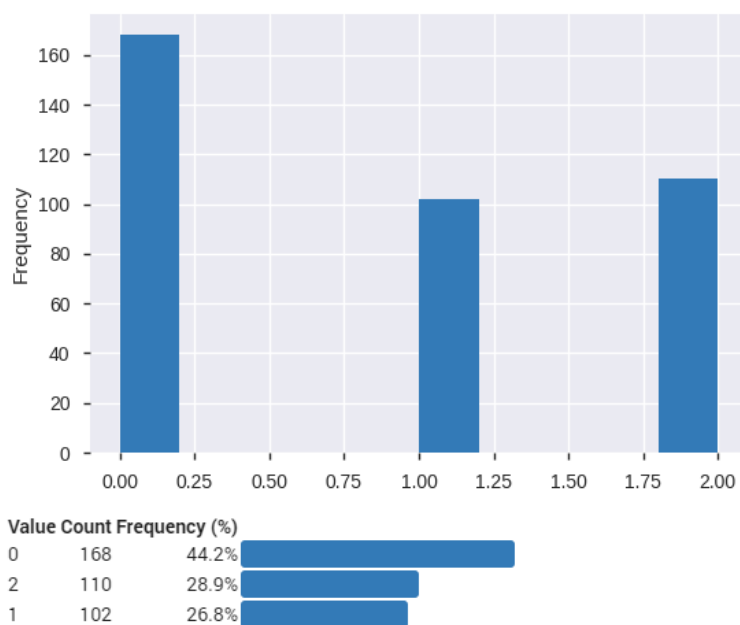


Figura 29- Análise da coluna VENC(Resultado final)

Fonte: Próprio autor

A imagem acima analisa o campo VENC, e revela que em mais de 40% dos jogos, quem vence é o time da casa.

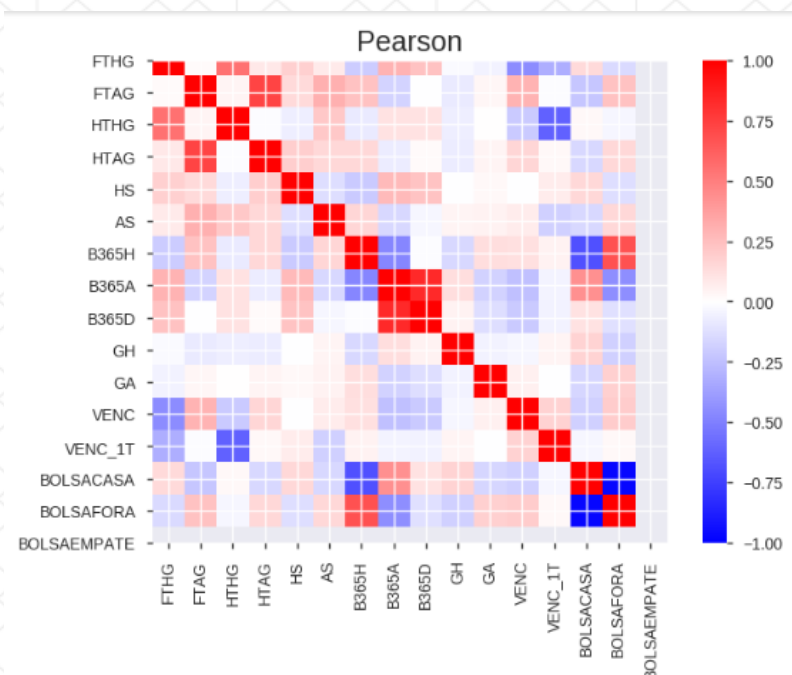


Figura 30-Correlação entre os campos

Fonte: Próprio autor

A figura 30 acima analisa a correlação entre os atributos, onde se percebe, por exemplo, uma forte correlação entre o vencedor do primeiro tempo (VENC_1T) e o número de gols do primeiro tempo,

RESULTADOS

12. Descrição e análise dos resultados alcançados:

- a. Resultados positivos encontrados (caso existam). Explique;
- b. Resultados negativos encontrados (caso existam). Explique.

Um dos principais pontos positivos encontrados no trabalho foi o teste exaustivo para encontrar a melhor forma pra se obter uma boa precisão, visto que teve, ao longo do processo, mudanças de mindset e até mesmo mudanças de algoritmos.

A originalidade do trabalho também foi um ponto positivo, tanto com relação ao contexto do negócio, onde foi constatado que não existe nenhum produto parecido ainda, e também com relação à busca contínua por melhorias, onde foram criadas novas features como solução que impulsionaram muito a qualidade do algoritmo.

O principal resultado negativo foi a baixa acurácia, de modo geral do algoritmo.

CONCLUSÃO

13. Apresentação da conclusão:

- a. Principais contribuições que seu projeto gera aos envolvidos;
- b. Inovações, particularidades ou vantagens que o projeto/resultado possui em relação a similares;
- c. Limitações do projeto;
- d. Próximos passos necessários para que o projeto evolua/se desenvolva.

A maior contribuição é mostrar aos envolvidos que é possível realizar previsões de resultados de partida de futebol utilizando técnicas de machine learning, com uma qualidade aceitável aos que é conhecido no meio como "ODD de valor", que precisa ser maior que 50%.

A maior inovação que o projeto trará quando implementado será de fornecer apoio aos traders esportivos, com o respaldo de histórico recente sobre as partidas, eliminando assim, muito do achismo

As principais limitações são:

Os próximos passos do projeto serão:

- a criação de uma API para realização das previsões em tempo real,
- testes com outros algoritmos de classificação,
- a implementação para pegar os dados do dataset em tempo real;
- criação de aplicativo mobile para consumo dos serviços
- utilização de dados do twitter para fazer análise de sentimentos durante os jogos e fazer as estimativas

Na figura 31 abaixo segue um protótipo do produto a ser implementado, onde o usuário escolhe o campeonato, e entra com input de dados:

Escolha a data dos jogos

<

January 2017

>

Mo	Tu	We	Th	Fr	Sa	Su
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				

▼ La liga

Real Madrid x La Coruna

Barcelona x Espanyol

▼ Copa Itália

Milan x Inter

Juventus x Frosinone

Data: 20/12/2019

Jogo: Barcelona x Espanyol

Chutes a gol

Anfitrião

Visitante

Gols

Anfitrião

Visitante

Obter previsão

A previsão para este jogo é:

Figura 31-Interface para o usuário

Fonte: Próprio autor