UFC - Fight historical data from 1993 to 2019

Compilado das lutas de UFC, estatísticas dos lutadores e informações

Criado em 1993 como uma organização profissional de artes marciais mistas (MMA), o UFC® revolucionou a indústria da luta e hoje se destaca tanto como uma marca global premium de esporte quanto como uma empresa de produção de conteúdo e o maior provedor de eventos Pay-Per-View (PPV) do mundo.

O UFC segue uma história e uma tradição ricas de MMA competitivo que remonta ao Pancrácio, uma luta introduzida nos Jogos Olímpicos gregos no ano de 648 a.C. Há cerca de 80 anos, uma forma brasileira de MMA conhecida como Vale-Tudo despertou o interesse local pelo esporte. O UFC então introduziu o MMA organizado e sancionado nos Estados Unidos.

O objetivo era encontrar o "campeão supremo de luta" (Ultimate Fighting Champion) organizando um torneio de uma noite com os melhores atletas das diversas modalidades de artes marciais, Incluindo caratê, jiu-jítsu, boxe, kickboxing, grappling, wrestling, sumô e outros esportes de combate. O vencedor do torneio seria coroado o campeão.

Analisando arquivo raw\_fighter\_details.csv

Excluindo registros sem altura e/ou sem peso

Analisando arquivo data.csv

falar das variáveis

analise exploratória

Substituindo campos vazios por Nan, convertendo a data para datetime, e arrendondando tudo para 3 casas decimais

Cruzando dados de Peso, Altura e Envergadura entre as bases.

Vimos que os dados faltantes dos lutadores no dataset de lutas também não constam no dataset de lutadores

Divisões de peso

O UFC se divide nessas categorias de pesos:

Peso Palha (Strawweight) - até 52,2 kg /115 lb (Feminino)

Peso Mosca (Flyweight) - até 56,7 kg / 125 lb (Masculino e Feminino)

Peso Galo (Bantamweight) - até 61,2 kg / 135 lb (Masculino e Feminino)

Peso Pena (Featherweight) - até 65,8 kg / 145 lb (Masculino e Feminino)

Peso Leve (Lightweight) - até 70,3 kg / 155 lb

Peso Meio-Médio (Welterweight) - até 77,1 kg / 170 lb

Peso Médio (Middleweight) - até 83,9 kg / 185 lb

Peso Meio-Pesado (Light Heavyweight) - até 92,9 kg / 205 lb

Peso Pesado (Heavyweight) - até 120,2 kg / 265 lb

No começo do UFC não havia divisão de categorias por peso e o combate acontece em um round único sem tempo. Na base de dados, essas lutas receberam a nomeclatura de Open Weight e Catch Weight. Para a criação de um melhor modelo, essas categorias serão removidas da base.

Verificou-se que nem sempre haviam vencedores nas lutas. Também haviam empates.

Removendo os empates em Winner, e deletando as colunas R\_draw e B\_draw.

Verificando os estilos de lutadores.

Plotando o gráfico dos vencedores por cor de luva.

Vemos no gráfico acima que a cor é relevante!

Baseline: Red = 67,85%

Criando dois dataframes com os dados dos lutadores de luvas Azuis e Vermelhas.

Cada quadrado mostra a correlação entre as variáveis ​​em cada eixo. Valores mais próximos de zero significam que não há tendência linear entre as duas variáveis. Quanto mais próxima de 1 a correlação é, mais positivamente correlacionados eles são; isto é, à medida que um aumenta, o outro aumenta e quanto mais próximo de 1, mais forte é essa relação. Uma correlação mais próxima de -1 é semelhante, mas em vez de aumentar, uma variável diminuirá à medida que a outra aumenta. Quanto maior o número e mais escura a cor, maior é a correlação entre as duas variáveis.

When NaNs appear in the output but are not present in the inputs

Notice that all of the values in the column are identical:

ufc\_data['B\_color'] = 0

ufc\_data['R\_color'] = 1

If you look at the equations for Pearson's corr() you'll notice that both have a term in the denominator that subtracts the mean of y from each y-value. When each value of y is identical, the result is a vector of 0s. When you divide by zero, you get NaN. Another way of putting it, the standard deviation of x or y cannot be 0. When you have a vector of identical values, the std is 0.

The NaN, in this case, is interpretted as no correlation between the two variables. The correlation describes how much one variable changes as the other variable changes. That requires both variables to change.

Gráfico de distribuição da correlação de todas as váriáveis com a variável 'Winner'

Como vemos, a correlação de cada variável individualmente com a variável 'Winner' é baixa.

Lista com as variáveis de maior correlação com a variável 'Winner':

Criando dois dataframes com os dados dos lutadores de luvas Azuis e Vermelhas, para tentar identificar variáveis redundantes.

Para cada par de variáveis com correlação alta, podemos avaliar a possibilidade de manter só uma delas no modelo, para evitar ruídos.

Plotando o mapa de calor - Blue Fighter.

Maiores valores de correlação para cada variável - Blue Fighter:

Plotando o mapa de calor - Red Fighter.

Maiores valores de correlação para cada variável - Red Fighter:

Selecionando variáveis que julgamos mais relevantes, ou classificamos como importantes e eliminando redundâncias:

important = ['date',

'B\_fighter',

'R\_fighter',

'Winner',

'B\_avg\_DISTANCE\_att',

'B\_avg\_DISTANCE\_landed',

'B\_avg\_SIG\_STR\_att',

'B\_current\_lose\_streak',

'B\_current\_win\_streak',

'B\_longest\_win\_streak',

'B\_losses',

'B\_total\_rounds\_fought',

'B\_total\_title\_bouts',

'B\_win\_by\_Decision\_Majority',

'B\_win\_by\_Decision\_Split',

'B\_win\_by\_Decision\_Unanimous',

'B\_win\_by\_KO/TKO',

'B\_win\_by\_Submission',

'B\_win\_by\_TKO\_Doctor\_Stoppage',

'B\_wins',

'B\_Stance',

'B\_Height\_cms',

'B\_Reach\_cms',

'B\_Weight\_lbs',

'R\_avg\_DISTANCE\_att',

'R\_avg\_DISTANCE\_landed',

'R\_avg\_SIG\_STR\_att',

'R\_current\_lose\_streak',

'R\_current\_win\_streak',

'R\_longest\_win\_streak',

'R\_losses',

'R\_total\_rounds\_fought',

'R\_total\_title\_bouts',

'R\_win\_by\_Decision\_Majority',

'R\_win\_by\_Decision\_Split',

'R\_win\_by\_Decision\_Unanimous',

'R\_win\_by\_KO/TKO',

'R\_win\_by\_Submission',

'R\_win\_by\_TKO\_Doctor\_Stoppage',

'R\_wins',

'R\_Stance',

'R\_Height\_cms',

'R\_Reach\_cms',

'R\_Weight\_lbs',

'B\_age',

'R\_age']

Plotando os mapas de calor com dados reduzidos e mais importantes, assim como as tabelas de correlação.

Pairplot:

Obtendo o percentual de dados faltantes.

Dataset original:

% Missing in 109 Features:

Referee 0.45

B\_avg\_BODY\_att 23.90

B\_avg\_BODY\_landed 23.90

B\_avg\_CLINCH\_att 23.90

B\_avg\_CLINCH\_landed 23.90

B\_avg\_DISTANCE\_att 23.90

B\_avg\_DISTANCE\_landed 23.90

B\_avg\_GROUND\_att 23.90

B\_avg\_GROUND\_landed 23.90

B\_avg\_HEAD\_att 23.90

B\_avg\_HEAD\_landed 23.90

B\_avg\_KD 23.90

B\_avg\_LEG\_att 23.90

B\_avg\_LEG\_landed 23.90

B\_avg\_PASS 23.90

B\_avg\_REV 23.90

B\_avg\_SIG\_STR\_att 23.90

B\_avg\_SIG\_STR\_landed 23.90

B\_avg\_SIG\_STR\_pct 23.90

B\_avg\_SUB\_ATT 23.90

B\_avg\_TD\_att 23.90

B\_avg\_TD\_landed 23.90

B\_avg\_TD\_pct 23.90

B\_avg\_TOTAL\_STR\_att 23.90

B\_avg\_TOTAL\_STR\_landed 23.90

B\_avg\_opp\_BODY\_att 23.90

B\_avg\_opp\_BODY\_landed 23.90

B\_avg\_opp\_CLINCH\_att 23.90

B\_avg\_opp\_CLINCH\_landed 23.90

B\_avg\_opp\_DISTANCE\_att 23.90

B\_avg\_opp\_DISTANCE\_landed 23.90

B\_avg\_opp\_GROUND\_att 23.90

B\_avg\_opp\_GROUND\_landed 23.90

B\_avg\_opp\_HEAD\_att 23.90

B\_avg\_opp\_HEAD\_landed 23.90

B\_avg\_opp\_KD 23.90

B\_avg\_opp\_LEG\_att 23.90

B\_avg\_opp\_LEG\_landed 23.90

B\_avg\_opp\_PASS 23.90

B\_avg\_opp\_REV 23.90

B\_avg\_opp\_SIG\_STR\_att 23.90

B\_avg\_opp\_SIG\_STR\_landed 23.90

B\_avg\_opp\_SIG\_STR\_pct 23.90

B\_avg\_opp\_SUB\_ATT 23.90

B\_avg\_opp\_TD\_att 23.90

B\_avg\_opp\_TD\_landed 23.90

B\_avg\_opp\_TD\_pct 23.90

B\_avg\_opp\_TOTAL\_STR\_att 23.90

B\_avg\_opp\_TOTAL\_STR\_landed 23.90

B\_total\_time\_fought(seconds) 23.90

B\_Stance 2.84

B\_Height\_cms 0.12

B\_Reach\_cms 11.13

B\_Weight\_lbs 0.08

R\_avg\_BODY\_att 12.18

R\_avg\_BODY\_landed 12.18

R\_avg\_CLINCH\_att 12.18

R\_avg\_CLINCH\_landed 12.18

R\_avg\_DISTANCE\_att 12.18

R\_avg\_DISTANCE\_landed 12.18

R\_avg\_GROUND\_att 12.18

R\_avg\_GROUND\_landed 12.18

R\_avg\_HEAD\_att 12.18

R\_avg\_HEAD\_landed 12.18

R\_avg\_KD 12.18

R\_avg\_LEG\_att 12.18

R\_avg\_LEG\_landed 12.18

R\_avg\_PASS 12.18

R\_avg\_REV 12.18

R\_avg\_SIG\_STR\_att 12.18

R\_avg\_SIG\_STR\_landed 12.18

R\_avg\_SIG\_STR\_pct 12.18

R\_avg\_SUB\_ATT 12.18

R\_avg\_TD\_att 12.18

R\_avg\_TD\_landed 12.18

R\_avg\_TD\_pct 12.18

R\_avg\_TOTAL\_STR\_att 12.18

R\_avg\_TOTAL\_STR\_landed 12.18

R\_avg\_opp\_BODY\_att 12.18

R\_avg\_opp\_BODY\_landed 12.18

R\_avg\_opp\_CLINCH\_att 12.18

R\_avg\_opp\_CLINCH\_landed 12.18

R\_avg\_opp\_DISTANCE\_att 12.18

R\_avg\_opp\_DISTANCE\_landed 12.18

R\_avg\_opp\_GROUND\_att 12.18

R\_avg\_opp\_GROUND\_landed 12.18

R\_avg\_opp\_HEAD\_att 12.18

R\_avg\_opp\_HEAD\_landed 12.18

R\_avg\_opp\_KD 12.18

R\_avg\_opp\_LEG\_att 12.18

R\_avg\_opp\_LEG\_landed 12.18

R\_avg\_opp\_PASS 12.18

R\_avg\_opp\_REV 12.18

R\_avg\_opp\_SIG\_STR\_att 12.18

R\_avg\_opp\_SIG\_STR\_landed 12.18

R\_avg\_opp\_SIG\_STR\_pct 12.18

R\_avg\_opp\_SUB\_ATT 12.18

R\_avg\_opp\_TD\_att 12.18

R\_avg\_opp\_TD\_landed 12.18

R\_avg\_opp\_TD\_pct 12.18

R\_avg\_opp\_TOTAL\_STR\_att 12.18

R\_avg\_opp\_TOTAL\_STR\_landed 12.18

R\_total\_time\_fought(seconds) 12.18

R\_Stance 2.57

R\_Height\_cms 0.06

R\_Reach\_cms 4.32

R\_Weight\_lbs 0.04

B\_age 1.99

R\_age 0.51

Dataset tratado:

% Missing in 16 Features:

B\_avg\_DISTANCE\_att 23.90

B\_avg\_DISTANCE\_landed 23.90

B\_avg\_SIG\_STR\_att 23.90

B\_Stance 2.84

B\_Height\_cms 0.12

B\_Reach\_cms 11.13

B\_Weight\_lbs 0.08

R\_avg\_DISTANCE\_att 12.18

R\_avg\_DISTANCE\_landed 12.18

R\_avg\_SIG\_STR\_att 12.18

R\_Stance 2.57

R\_Height\_cms 0.06

R\_Reach\_cms 4.32

R\_Weight\_lbs 0.04

B\_age 1.99

R\_age 0.51

Com a base de dados ufc\_raw\_fighter, vamos verificar os registros que não possuem envergadura (variável 'Reach').

Obtendo mapa de calor entre peso, altura, e envergadura.

Regressão Linear

Prevendo a envergadura utilizando regressão linear.

MAE: 3.9223285372879144

MSE: 24.649623775345397

RMSE: 4.964838746157361

Árvore de Decisão

Prevendo envergadura utilizando Árvore de Decisão

MAE: 3.841899631471346

MSE: 23.820057648825994

RMSE: 4.8805796427090495

Random Forest

MAE: 3.8323992962106703

MSE: 23.664332055990876

RMSE: 4.864599886526216

Redes Neurais

MAE: 3.5154778221330614

MSE: 21.376598997981255

RMSE: 4.623483426809407

SVM - kernel linear

MAE: 3.7203119292597755

MSE: 22.583527950641187

RMSE: 4.752212952997917

SVM - kernel rbf

MAE: 4.559574390784046

MSE: 40.61893806554289

RMSE: 6.373298836987239

Substituição de envergaduras NaN por valores previstos

O melhor resultado foi obtido com o modelo de Redes Neurais.

Substituindo as envergaduras (variável 'Reach') NaN pelos valores previstos:

Função para aplicar nvl no dataframe:

Inserindo os valores de envergadura previstos no dataframe ufc\_data - lutador B:

Inserindo os valores de envergadura previstos no dataframe ufc\_data - lutador R:

# Convertendo as colunas B\_Reach\_cms e R\_Reach\_cms para float:

Eliminando registros com valores vazios - Gráfico NaN:

Eliminando as linhas com valores nulos, para não dropar essas as colunas no próximo step:

df\_lutas = ufc\_data

df\_lutas = df\_lutas[pd.notna(df\_lutas['B\_Height\_cms'])].reset\_index(drop=True)

df\_lutas = df\_lutas[pd.notna(df\_lutas['R\_Height\_cms'])].reset\_index(drop=True)

​

df\_lutas = df\_lutas[pd.notna(df\_lutas['B\_Weight\_lbs'])].reset\_index(drop=True)

df\_lutas = df\_lutas[pd.notna(df\_lutas['R\_Weight\_lbs'])].reset\_index(drop=True)

​

df\_lutas = df\_lutas[pd.notna(df\_lutas['B\_age'])].reset\_index(drop=True)

df\_lutas = df\_lutas[pd.notna(df\_lutas['R\_age'])].reset\_index(drop=True)

​

df\_lutas = df\_lutas[pd.notna(df\_lutas['B\_Stance'])].reset\_index(drop=True)

df\_lutas = df\_lutas[pd.notna(df\_lutas['R\_Stance'])].reset\_index(drop=True)

​

Removemos então as variáveis abaixo, devido à grande quantide de registros com valores nulos:

'B\_avg\_DISTANCE\_att'

'B\_avg\_DISTANCE\_landed'

'B\_avg\_SIG\_STR\_att'

'R\_avg\_DISTANCE\_att'

'R\_avg\_DISTANCE\_landed'

'R\_avg\_SIG\_STR\_att'

Converte as colunas categóricas 'B\_Stance' e 'R\_Stance' em numéricas:

B\_Stance\_Open Stance B\_Stance\_Orthodox B\_Stance\_Sideways B\_Stance\_Southpaw B\_Stance\_Switch R\_Stance\_Open Stance R\_Stance\_Orthodox R\_Stance\_Southpaw R\_Stance\_Switch R\_Stance\_Sideways

Lista final de colunas que seguirão para predição:

['R\_fighter',

'B\_fighter',

'date',

'location',

'Winner',

'title\_bout',

'weight\_class',

'no\_of\_rounds',

'B\_current\_lose\_streak',

'B\_current\_win\_streak',

'B\_longest\_win\_streak',

'B\_losses',

'B\_total\_rounds\_fought',

'B\_total\_title\_bouts',

'B\_win\_by\_Decision\_Majority',

'B\_win\_by\_Decision\_Split',

'B\_win\_by\_Decision\_Unanimous',

'B\_win\_by\_KO/TKO',

'B\_win\_by\_Submission',

'B\_win\_by\_TKO\_Doctor\_Stoppage',

'B\_wins',

'B\_Height\_cms',

'B\_Reach\_cms',

'B\_Weight\_lbs',

'R\_current\_lose\_streak',

'R\_current\_win\_streak',

'R\_longest\_win\_streak',

'R\_losses',

'R\_total\_rounds\_fought',

'R\_total\_title\_bouts',

'R\_win\_by\_Decision\_Majority',

'R\_win\_by\_Decision\_Split',

'R\_win\_by\_Decision\_Unanimous',

'R\_win\_by\_KO/TKO',

'R\_win\_by\_Submission',

'R\_win\_by\_TKO\_Doctor\_Stoppage',

'R\_wins',

'R\_Height\_cms',

'R\_Reach\_cms',

'R\_Weight\_lbs',

'B\_age',

'R\_age',

'B\_color',

'R\_color',

'B\_Stance\_Open Stance',

'B\_Stance\_Orthodox',

'B\_Stance\_Sideways',

'B\_Stance\_Southpaw',

'B\_Stance\_Switch',

'R\_Stance\_Open Stance',

'R\_Stance\_Orthodox',

'R\_Stance\_Southpaw',

'R\_Stance\_Switch',

'R\_Stance\_Sideways']

Comparando essa lista, com a lista de variáveis que consideramos importantes na análise exploratória inicial e gráficos de correlação, ficamos muito satisfesitos com o resultado.

Substituindo True e False por 1 e 0 na coluna 'title\_bout':

INICIANDO PREVISÃO DE RESULTADOS DE LUTAS

Arvore de decisão

print(confusion\_matrix(y\_test,predictions))

[[125 185]

[192 416]]

print(classification\_report(y\_test,predictions))

precision recall f1-score support

0 0.39 0.40 0.40 310

1 0.69 0.68 0.69 608

accuracy 0.59 918

macro avg 0.54 0.54 0.54 918

weighted avg 0.59 0.59 0.59 918

Random Forest

print(confusion\_matrix(y\_test,rfc\_pred))

[[ 64 246]

[ 39 569]]

print(classification\_report(y\_test,rfc\_pred))

precision recall f1-score support

0 0.62 0.21 0.31 310

1 0.70 0.94 0.80 608

accuracy 0.69 918

macro avg 0.66 0.57 0.55 918

weighted avg 0.67 0.69 0.63 918

Naive Bayes - GaussianNB

print(confusion\_matrix(y\_test,gnb\_pred))

[[290 20]

[539 69]]

print(classification\_report(y\_test,gnb\_pred))

precision recall f1-score support

0 0.35 0.94 0.51 310

1 0.78 0.11 0.20 608

accuracy 0.39 918

macro avg 0.56 0.52 0.35 918

weighted avg 0.63 0.39 0.30 918

Naive Bayes - MultinomialNB

print(confusion\_matrix(y\_test,mnb\_pred))

[[150 160]

[191 417]]

print(classification\_report(y\_test,mnb\_pred))

precision recall f1-score support

0 0.44 0.48 0.46 310

1 0.72 0.69 0.70 608

accuracy 0.62 918

macro avg 0.58 0.58 0.58 918

weighted avg 0.63 0.62 0.62 918

Naive Bayes - BernoulliNB

print(confusion\_matrix(y\_test,bnb\_pred))

[[ 78 232]

[ 86 522]]

print(classification\_report(y\_test,bnb\_pred))

precision recall f1-score support

0 0.48 0.25 0.33 310

1 0.69 0.86 0.77 608

accuracy 0.65 918

macro avg 0.58 0.56 0.55 918

weighted avg 0.62 0.65 0.62 91

Naive Bayes - ComplementNB

print(confusion\_matrix(y\_test,cnb\_pred))

[[160 150]

[210 398]]

print(classification\_report(y\_test,cnb\_pred))

precision recall f1-score support

0 0.43 0.52 0.47 310

1 0.73 0.65 0.69 608

accuracy 0.61 918

macro avg 0.58 0.59 0.58 918

weighted avg 0.63 0.61 0.61 918

XGBoost

print(confusion\_matrix(y\_test,xgb\_pred))

[[100 210]

[112 496]]

print(classification\_report(y\_test,xgb\_pred))

precision recall f1-score support

0 0.47 0.32 0.38 310

1 0.70 0.82 0.75 608

accuracy 0.65 918

macro avg 0.59 0.57 0.57 918

weighted avg 0.62 0.65 0.63 918

Rede Neural

print(confusion\_matrix(y\_test, y\_pred))

[[ 64 246]

[ 34 574]]

p

print(classification\_report(y\_test, y\_pred))

precision recall f1-score support

0 0.65 0.21 0.31 310

1 0.70 0.94 0.80 608

accuracy 0.69 918

macro avg 0.68 0.58 0.56 918

weighted avg 0.68 0.69 0.64 918

Partindo de uma base desbalanceada, os melhores modelos (Random Forest e Rede Neural) atingiram acurácia de 69%, que não é um valor expressivo, visto que apenas a cor da luva representa um favoritismo de 67,85% para o vermelho (Baseline: Red = 67,85%). Além disso, também observamos forte desequilíbrio no recall

Nova abordagem forçando o balanceamento da base

Apagando 40% dos registros de vitória do vermelho

Filtra aleatoriamente 40% das lutas vencidas pelo vermelho e apaga:

XGBoost

print(confusion\_matrix(y\_test,xgb\_pred))

[[145 139]

[127 259]]

print(classification\_report(y\_test,xgb\_pred))

precision recall f1-score support

0 0.53 0.51 0.52 284

1 0.65 0.67 0.66 386

accuracy 0.60 670

macro avg 0.59 0.59 0.59 670

weighted avg 0.60 0.60 0.60 670

Random Forest

print(confusion\_matrix(y\_test,rfc\_pred))

[[146 138]

[120 266]]

print(classification\_report(y\_test,rfc\_pred))

precision recall f1-score support

0 0.55 0.51 0.53 284

1 0.66 0.69 0.67 386

accuracy 0.61 670

macro avg 0.60 0.60 0.60 670

weighted avg 0.61 0.61 0.61 670

Rede Neural

print(confusion\_matrix(y\_test, y\_pred))

[[168 116]

[144 242]]

print(classification\_report(y\_test, y\_pred))

precision recall f1-score support

0 0.54 0.59 0.56 284

1 0.68 0.63 0.65 386

accuracy 0.61 670

macro avg 0.61 0.61 0.61 670

weighted avg 0.62 0.61 0.61 670

Eliminando cerca de 1600 vitórias do vermelho (redução considerável no tamanho da amostra), o recall do azul ficou muito alto, provavelmente porque os atributos do azul são geralmente piores. Como foram pegas todas as lutas que o azul ganhou e eliminadas 40% das lutas do vermelho para forçar o balanceamento da base, o resultado obtido não foi satisfatório.

Invertendo as cores em 25% das lutas vencidas pelo vermelho

Filtra aleatoriamente 25% das lutas vencidas pelo vermelho e inverte as cores:

Com a inversão das cores em 25% das lutas vencidas pelo vermelho, obtivemos um dataset balanceado, sem perda de registros.

XGBoost

print(confusion\_matrix(y\_test,xgb\_pred))

[[281 168]

[201 268]]

print(classification\_report(y\_test,xgb\_pred))

precision recall f1-score support

0 0.58 0.63 0.60 449

1 0.61 0.57 0.59 469

accuracy 0.60 918

macro avg 0.60 0.60 0.60 918

weighted avg 0.60 0.60 0.60 91

Random Forest

print(confusion\_matrix(y\_test,rfc\_pred))

[[264 185]

[202 267]]

print(classification\_report(y\_test,rfc\_pred))

precision recall f1-score support

0 0.57 0.59 0.58 449

1 0.59 0.57 0.58 469

accuracy 0.58 918

macro avg 0.58 0.58 0.58 918

weighted avg 0.58 0.58 0.58 918

Rede Neural

print(confusion\_matrix(y\_test, y\_pred))

[[273 176]

[174 295]]

print(classification\_report(y\_test, y\_pred))

precision recall f1-score support

0 0.61 0.61 0.61 449

1 0.63 0.63 0.63 469

accuracy 0.62 918

macro avg 0.62 0.62 0.62 918

weighted avg 0.62 0.62 0.62 918

Com a inversão das cores em 25% das lutas vencidas pelo vermelho, a Rede Neural apresentou o melhor resultado, com valores equilibrados de precision e recall, além de uma acurácia de 62% sobre uma base balanceada.

Obtendo as probabilidades, para visualização do resultado final:

Visualização do resultado final:

Conferindo a acurácia do modelo:

Resultado\_Previsto

df\_acertos=df[df['Resultado\_Previsto']==df['Resultado\_Real']]

len(df\_acertos.index)/len(df)

0.6187363834422658

Precisão do modelo para vitória do vermelho:

df\_vitorias\_Red=df[df['Resultado\_Real']==1]

df\_acertos\_Red=df\_vitorias\_Red[df\_vitorias\_Red['Resultado\_Previsto']==1]

len(df\_acertos\_Red.index)/len(df\_vitorias\_Red.index)

0.6289978678038379

Precisão do modelo para vitória do azul:

df\_vitorias\_Blue=df[df['Resultado\_Real']==0]

df\_acertos\_Blue=df\_vitorias\_Blue[df\_vitorias\_Blue['Resultado\_Previsto']==0]

len(df\_acertos\_Blue.index)/len(df\_vitorias\_Blue.index)

0.6080178173719376

Jogando com as probabilidades:

Precisão do modelo para vitória do vermelho, quando a probabilidade for maior que 75%:

5

df\_prob\_Red = df[df['Prob\_Win\_Red']>=.75]

df\_vit\_Red=df\_prob\_Red[df\_prob\_Red['Resultado\_Real']==1]

len(df\_vit\_Red.index)/len(df\_prob\_Red.index)

0.8125

Precisão do modelo para vitória do azul, quando a probabilidade for maior que 75%:

#

df\_prob\_Blue = df[df['Prob\_Win\_Blue']>=.75]

df\_vit\_Blue=df\_prob\_Blue[df\_prob\_Blue['Resultado\_Real']==0]

len(df\_vit\_Blue.index)/len(df\_prob\_Blue.index)

#df\_prob\_Blue

0.8292682926829268

Jogando com as probabilidades previstas pelo modelo, conseguimos chegar a resultados ainda mais satisfatórios.

Modelando previsões por categoria de peso

df\_lutas = df\_final

weight\_class=df\_lutas.groupby('weight\_class')['weight\_class'].count().reset\_index(name="Qtd")

weight\_class

weight\_class Qtd

0 Bantamweight 340

1 Featherweight 411

2 Flyweight 182

3 Heavyweight 455

4 Light Heavyweight 474

5 Lightweight 894

6 Middleweight 681

7 Welterweight 914

8 Women's Bantamweight 92

9 Women's Featherweight 8

10 Women's Flyweight 32

11 Women's Strawweight 106

Bantamweight

[[22 16]

[21 9]]

precision recall f1-score support

0 0.51 0.58 0.54 38

1 0.36 0.30 0.33 30

accuracy 0.46 68

macro avg 0.44 0.44 0.44 68

weighted avg 0.44 0.46 0.45 68

Featherweight

[[35 19]

[18 11]]

precision recall f1-score support

0 0.66 0.65 0.65 54

1 0.37 0.38 0.37 29

accuracy 0.55 83

macro avg 0.51 0.51 0.51 83

weighted avg 0.56 0.55 0.56 83

Flyweight

[[15 9]

[ 5 8]]

precision recall f1-score support

0 0.75 0.62 0.68 24

1 0.47 0.62 0.53 13

accuracy 0.62 37

macro avg 0.61 0.62 0.61 37

weighted avg 0.65 0.62 0.63 37

Heavyweight

[[25 27]

[14 25]]

precision recall f1-score support

0 0.64 0.48 0.55 52

1 0.48 0.64 0.55 39

accuracy 0.55 91

macro avg 0.56 0.56 0.55 91

weighted avg 0.57 0.55 0.55 91

Light Heavyweight

[[20 25]

[16 34]]

precision recall f1-score support

0 0.56 0.44 0.49 45

1 0.58 0.68 0.62 50

accuracy 0.57 95

macro avg 0.57 0.56 0.56 95

weighted avg 0.57 0.57 0.56 95

Lightweight

[[37 56]

[22 64]]

precision recall f1-score support

0 0.63 0.40 0.49 93

1 0.53 0.74 0.62 86

accuracy 0.56 179

macro avg 0.58 0.57 0.55 179

weighted avg 0.58 0.56 0.55 179

Middleweight

[[11 51]

[20 55]]

precision recall f1-score support

0 0.35 0.18 0.24 62

1 0.52 0.73 0.61 75

accuracy 0.48 137

macro avg 0.44 0.46 0.42 137

weighted avg 0.44 0.48 0.44 137

Welterweight

[[77 16]

[57 33]]

precision recall f1-score support

0 0.57 0.83 0.68 93

1 0.67 0.37 0.47 90

accuracy 0.60 183

macro avg 0.62 0.60 0.58 183

weighted avg 0.62 0.60 0.58 183

Women's Bantamweight

[[5 4]

[7 3]]

precision recall f1-score support

0 0.42 0.56 0.48 9

1 0.43 0.30 0.35 10

accuracy 0.42 19

macro avg 0.42 0.43 0.41 19

weighted avg 0.42 0.42 0.41 19

Women's Featherweight

[[0 1]

[0 1]]

precision recall f1-score support

0 0.00 0.00 0.00 1

1 0.50 1.00 0.67 1

accuracy 0.50 2

macro avg 0.25 0.50 0.33 2

weighted avg 0.25 0.50 0.33 2

Women's Flyweight

[[2 0]

[4 1]]

precision recall f1-score support

0 0.33 1.00 0.50 2

1 1.00 0.20 0.33 5

accuracy 0.43 7

macro avg 0.67 0.60 0.42 7

weighted avg 0.81 0.43 0.38 7

Women's Strawweight

[[10 2]

[ 8 2]]

precision recall f1-score support

0 0.56 0.83 0.67 12

1 0.50 0.20 0.29 10

accuracy 0.55 22

macro avg 0.53 0.52 0.48 22

weighted avg 0.53 0.55 0.49 22

Os resultados por categoria de peso não são expressivos, por serem amostras com poucos registros.