UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ARTES, CIÊNCIAS E HUMANIDADES

ADA MARIA CAYRES FERNANDEZ
BEATRIZ THOMPSON SATHLER FREITAS
LEONARDO COPPOLA BIAZUCCI
MICHELLY RODRIGUES DA SILVA
THIAGO DE OLIVEIRA DEODATO
RAFAEL MARTINS CAMPOS

T2 - TÉCNICA DE DEPENDÊNCIA: REGRESSÃO LINEAR

SÃO PAULO - SP
2021

SUMÁRIO

1. INTRODUÇÃO

Será feito neste trabalho o uso da técnica de dependência conhecida como regressão linear sobre um recorte de banco de dados que diz respeito a partidas jogadas em *League of Legends*, jogo online famoso mundialmente. O uso da regressão linear nos permitirá descobrir ao final da análise se é possível prever o número de mortes do time azul a partir das variáveis explicativas e descobrir o quanto exatamente uma das equipes está prejudicada no jogo com base no número de mortes do time.

A análise será feita utilizando códigos na linguagem *Python* com auxílio das bibliotecas *pandas*, *numpy*, *matplotlib*, *pylab*, *scipy*, *seaborn* e *sklearn*. A execução dos códigos será realizada no *Google Colaboratory* (ou simplesmente *Colab*).

2. OBJETIVOS

Este documento tem como objetivo principal o estudo da relação entre as variáveis do banco de dados dos primeiros 10 minutos das partidas do jogo online conhecido mundialmente, *League of Legends*. Com o uso da técnica da regressão linear, é possível definir quais atributos geram mais vantagem ou desvantagem para uma das equipes. Esta análise é de profunda importância quando se está em um ambiente competitivo, pois ao saber o que lhe deixará na frente dos seus inimigos, o jogador pode guiar suas ações de acordo com o que foi encontrado na análise para ficar sempre na frente dos outros competidores.

Essas informações geram o "meta" do jogo. O “meta” é o padrão de ações mais eficiente e que deverá ser seguido durante a partida. Sendo assim, ao se utilizar de técnicas cada vez mais profundas de análise, o jogo passa a ficar cada vez mais complexo, e o conhecimento cada vez mais difícil de ser obtido. Desta forma, podemos dizer que a obtenção desse conhecimento é ainda mais eficaz, pois não é uma informação tão simples de ser obtida pelo jogador comum.

Neste documento, temos como foco descobrir se é possível prever o número de mortes do time azul a partir das variáveis explicativas. Como um número de mortes alto

normalmente está associado ao time que está atrás, e ao realizarmos tal análise, podemos descobrir o quanto exatamente uma das equipes está prejudicada no jogo.

3. PRÉ-PROCESSAMENTO

Antes de iniciar a análise, criamos o *data frame* (recorte do banco de dados) apenas com as cinco colunas de interesse. Em seguida, traduzimos do inglês para o português do Brasil o nome das cinco variáveis envolvidas na análise.

```
url_lol = 'https://raw.githubusercontent.com/michellyrds/mqaa/master/datasets/high_diamond_ranked_10min.csv'

url = url_lol
raw_data = pd.read_csv(url)

df_raw = raw_data.filter(['blueWardsPlaced', 'blueKills', 'blueDeaths', 'blueEliteMonsters', 'blueGoldDiff'], axis=1)

df = df_raw.rename(columns={'blueWardsPlaced' : 'azulSentinelas',
                            'blueKills' : 'azulAbates',
                            'blueDeaths' : 'azulMortes',
                            'blueEliteMonsters' : 'azulMonstrosEpicos',
                            'blueGoldDiff' : 'azulDiferencaOuros'})
```

Figura 1: linhas de código do pré-processamento do banco de dados.

.

```
df.head(10)
```

	azulSentinelas	azulAbates	azulMortes	azulMonstrosEpicos	azulDiferencaOuros
0	28	9	6	0	643
1	12	5	5	0	-2908
2	15	7	11	1	-1172
3	43	4	5	1	-1321
4	75	6	6	0	-1004
5	18	5	3	1	698
6	18	7	6	1	2411
7	16	5	13	0	-2615
8	16	7	7	0	-1979
9	13	4	5	1	-1548

Figura 2: mostrando as 10 primeiras linhas do *data frame* traduzido.

4. APLICAÇÃO DA REGRESSÃO LINEAR

Começamos escolhendo a variável quantitativa de interesse como a *‘azulMortes’* que contabiliza as mortes sofridas pelo time Azul.

As variáveis explicativas, por sua vez, são as seguintes:

- *‘azulSentinelas’* (sentinelas posicionadas pelo time Azul);
- *‘azulAbates’* (abates realizados pelo time Azul);
- *‘azulMonstrosEpicos’* (monstros épicos derrotados pelo time Azul);
- *‘azulDiferencaOuros’* (diferença de ouros do time Azul).

Separamos o *dataset* em dois conjuntos: Treinamento e Teste. Depois separamos o *dataset* de treinamento nas variáveis explicativas da variável aleatória de interesse (*‘azulMortes’*).

```
train, test = train_test_split(df, test_size=0.2)

x = train.filter(['azulSentinelas', 'azulAbates', 'azulMonstrosEpicos', 'azulDiferencaOuros'])
y = train.filter(['azulMortes'])
```

Figura 3: linhas de código da separação em dois conjuntos.

```
y.head(10)
```

	azulMortes
1247	3
4676	5
6676	2
4491	3
4233	9
2734	6
6074	10
2643	5
8545	7
7216	8

Figura 4: mostrando as 10 primeiras linhas do grupo separado y.

Criamos o modelo da regressão linear múltipla.

```
model = LinearRegression().fit(x, y)
```

Figura 5: linha de código de criação do modelo da regressão linear múltipla.

O modelo possui os seguintes pesos:

```
print(model.coef_)
```

```
[[ 0.00089959  0.7194373  -0.1069605  -0.00134058]]
```

Figura 6: linha de código que mostra os pesos das variáveis, sendo da esquerda para a direita: ‘azulSentinelas’, ‘azulAbates’, azulMonstrosEpicos‘, ‘azulDiferencaOuros’.’

O modelo possui o seguinte resíduo:

```
print(model.intercept_)
```

```
[1.73574819]
```

Figura 7: linha de código que mostra o resíduo.

Fazendo com que a sua equação seja igual a:

$$\mathbf{Y = 0.00089959X_1 + 0.7194373X_2 - 0.1069605X_3 - 0.00134058X_4 + 1.73574819}$$

Utilizando quatro variáveis (*'azulSentinelas'*, *'azulAbates'*, *'azulMonstrosEpicos'* e *'azulDiferencaOuros*') para estimar o número de mortes do time azul (*‘azulMortes’*), foram realizados testes com duas amostras do *data frame*:

```
x_to_pred = pd.DataFrame([28, 9, 0, 643]).T

y = 6
y_pred = model.predict(x_to_pred)

print("Valor real: {} \nValor predito pelo modelo: {}".format(y,y_pred))
```

```
Valor real: 6
Valor predito pelo modelo: [[7.37157228]]
```

Figura 8: linhas de código da criação da primeira amostra do *data frame* e da realização do teste;

```
x_to_pred_2 = pd.DataFrame([16, 5,0 ,-2615]).T

y = 13
y_pred = model.predict(x_to_pred_2)

print("Valor real: {} \nValor predito pelo modelo: {}".format(y,y_pred))
```

```
Valor real: 13
Valor predito pelo modelo: [[8.84008907]]
```

Figura 9: linhas de código da criação da segunda amostra do *data frame* e da realização do teste;

Criou-se o *array* de predição do conjunto de testes e depois comparamos com o *array* da variável aleatória de interesse, identificando uma similaridade.

```
x_test = test.filter(['azulSentinelas', 'azulAbates', 'azulMonstrosEpicos', 'azulDiferencaOuros'])
y_test = test.filter(['azulMortes'])
```

Figura 10: linhas de código da criação do *array* de predição e do *array* da variável aleatória de interesse.

```
y_test_pred = pd.DataFrame(model.predict(x_test))
```

```
y_test_pred.head(10)
```

	0
0	6.717345
1	7.275415
2	5.710397
3	7.681482
4	5.600924
5	8.160535
6	5.942945
7	5.322840
8	4.321310
9	9.178826

```
y_test.head(10)
```

	azulMortes
7068	6
3786	5
26	6
367	8
7297	4
707	9
282	5
176	5
260	5
7224	9

Figura 11: linhas de código para exibir 10 linhas de cada array e a comparação dos resultados obtidos entre o array de predição e o array da variável de interesse.

Para se obter o nível de qualidade do ajustamento, realizou-se primeiramente o coeficiente de determinação e obteve-se o valor 0.721408989147467.

```
r_sq = model.score(x, y)
print("Coeficiente de determinação: {}".format(r_sq))
```

```
Coeficiente de determinação: 0.721408989147467
```

Figura 12: linhas de código para calcular o coeficiente de determinação e o seu resultado logo abaixo.

Para analisar a correlação, foi calculado o coeficiente de correlação.

```
correlation_coefficient = np.corrcoef(x)
print("Coeficiente de correlação: \n{}".format(correlation_coefficient))
```

```
Coeficiente de correlação:
[[ 1.         -0.98225498 -0.97687203 ...  0.98432402 -0.98198531
  -0.98265932]
 [-0.98225498  1.          0.99964011 ... -0.99993512  0.99999804
   0.99999755]
 [-0.97687203  0.99964011  1.         ... -0.99927078  0.99967546
   0.99957919]
 ...
 [ 0.98432402 -0.99993512 -0.99927078 ...  1.         -0.99991601
  -0.99995703]
 [-0.98198531  0.99999804  0.99967546 ... -0.99991601  1.
   0.99999311]
 [-0.98265932  0.99999755  0.99957919 ... -0.99995703  0.99999311
   1.        ]]
```

Figura 13: linhas de código para calcular o coeficiente de correlação e o seu resultado logo abaixo.

Além disso, obtivemos o coeficientes de correlação múltiplo na matriz:

```
corr = df.corr()
corr.style.background_gradient(cmap='coolwarm')
```

Figura 14: linhas de código para criar graficamente a matriz de correlação com os coeficientes de correlação múltiplo.

	azulSentinelas	azulAbates	azulMortes	azulMonstrosEpicos	azulDiferencaOuros
azulSentinelas	1.000000	0.018138	-0.002612	0.019892	0.015800
azulAbates	0.018138	1.000000	0.004044	0.178540	0.654148
azulMortes	-0.002612	0.004044	1.000000	-0.204764	-0.640000
azulMonstrosEpicos	0.019892	0.178540	-0.204764	1.000000	0.281464
azulDiferencaOuros	0.015800	0.654148	-0.640000	0.281464	1.000000

Figura 15: matriz de correlação com os coeficientes de correlação múltiplo.

No seguinte gráfico é possível entender o nível de correlação entre as variáveis baseando se no número do coeficiente de correlação e as cores do esquema.

```
corr = df.corr()
sns.heatmap(corr,
            xticklabels=corr.columns.values,
            yticklabels=corr.columns.values)
```

Figura 16: código para produzir o gráfico de correlação com base nos coeficientes de correlação.

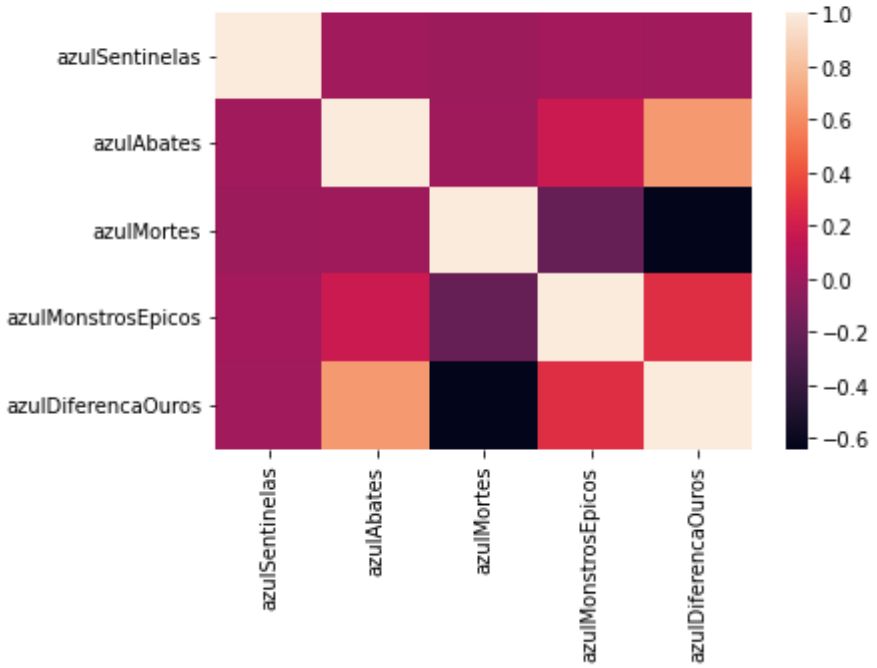

Figura 17: gráfico de correlação com base nos coeficientes de correlação.

5. INTERPRETAÇÃO DOS RESULTADOS

É possível observar alguns pontos nos resultados obtidos pela regressão linear múltipla. Com a definição da variável quantitativa de interesse e as variáveis explicativas, pela execução de modelo de regressão linear nas duas amostras do *data frame* utilizando as 4 variáveis (‘*azulSentinelas’*, *‘azulAbates’*, *‘azulMonstrosEpicos’* e ‘*azulDiferencaOuros’*) para prever o numero de mortes do time azul foi possível verificar que em um dos testes o resultado foi próximo do real (valor real 6, enquanto que o valor predito pelo modelo foi próximo a 7.3842). No segundo teste o valor predito teve uma distância maior, sendo o valor real 13 tendo o valor obtido próxima a 8,853. Porém, quando observamos uma amostra maior do conjunto de predições ao lado do conjunto de testes, pôde-se verificar a proximidade entre os resultados preditos pelo modelo e o conjunto de testes.

Um forte indicador de que nosso modelo é um bom modelo é o coeficiente de determinação, que para o nosso modelo de regressão possui um valor de aproximadamente 0,72387.

Outro ponto interessante que também podemos verificar é a matriz dos coeficientes de correlação que temos uma maior clareza de quais variáveis explicativas que possuem uma correlação maior com a variável quantitativa de interesse (‘*azulMortes’*) são a ‘*azulDiferencaOuros’* (-0,204764) e a ‘*azulMonstrosEpicos’* (-0,64).

6. CONCLUSÃO

Como apresentado na interpretação dos resultados, nós conseguimos identificar qual dos fatores escolhidos é o que mais está relacionado à quantidade de mortes da equipe azul, que nesse caso, foi o ‘*azulDiferencaOuros’*. Também foi possível observar o impacto das outras variáveis, que não se demonstraram tão expressivas, com a exceção da ‘*azulAbates’*, que também demonstrou um grande impacto.

Desta forma, o objetivo proposto no documento foi cumprido, e esta análise pôde ser aplicada dentro de jogo. Pelos resultados, obter uma vantagem de *golds* (ouros, em inglês) e procurar abater outros jogadores, devem ser o foco dos jogadores

nos 10 primeiros minutos de partida, visto que esses dois pontos apontados são os que mais impactaram no número de mortes.

7. REFERÊNCIAS

1. STOJILJKOVIĆ, Mirko. **Linear Regression in Python**. [*S. l.*], 2019. Disponível em: https://realpython.com/linear-regression-in-python/#multiple-linear-regression. Acesso em: 3 out. 2021.

2. SAEED, Mehreen. **Calculating Pearson Correlation Coefficient in Python with Numpy**. [*S. l.*], 2021. Disponível em: https://stackabuse.com/calculating-pearson-correlation-coefficient-in-python-with-numpy Acesso em: 3 out. 2021.

3. BRYDON, Michael. **Basic Analytics in Python**: Correlation and Scatterplots. [*S. l.*], 2021. Disponível em: https://www.sfu.ca/~mjbrydon/tutorials/BAinPy/08_correlation.html Acesso em: 3 out. 2021.

4. MAKLIN, Cory. **Least Squares Linear Regression In Python**. [*S. l.*], 16 ago. 2019. Disponível em: https://towardsdatascience.com/least-squares-linear-regression-in-python-54b87fc49e77. Acesso em: 3 out. 2021.

5. GOOGLE. **Google Colaboratory** - Disponível em: https://colab.research.google.com. Acesso em: 4 out. 2021.