Entrega Final: Proyecto Final

Jose Andres Echavarria Rios
Juan Jose Ospina Ramírez
David Camilo García Echavarria

Docente: Raúl Ramos

Universidad de Antioquia
Facultad de ingeniería

Medellín 12 de Noviembre del 2022
Antioquia

Objetivos:
● Evaluar el proceso que llevamos hasta el momento en el desarrollo del proyecto
● Identificar, evaluar y definir una ruta para la realización del proyecto
● Mostrar algunos notebooks con los avances hasta el momento El dataset usado es: https://www.kaggle.com/datasets/paololol/league-of-legends-ranked-matches

Marco teórico: League of Legends: League of Legends, es un videojuego multijugador de arena de batalla en línea desarrollado y publicado por Riot Games donde compiten dos equipos conformados por 5 jugadores cada uno con el objetivo de destruir un elemento en la base enemiga llamado “nexo”. El juego cuenta con una amplia variedad de personajes seleccionables llamados “Campeones” que se escogen según rol o utilidad.

Desarrollo del proyecto
En el dataset contiene un conjunto de datos de las partidas clasificatorias del juego (Rankeds).

Para esta sección del proyecto lo que se busca hacer principalmente es la lectura y análisis de los datos, a su vez de la limpieza de estos para que el dataset sea más amigable de trabajar, también se estimó quién es favorable para cada enfrentamiento con el objetivo de predecir quién está más capacitado para ganar, para esto se realizó un notebook donde esta este proceso documentado en forma de código.

Nuestro siguiente objetivo será: teniendo una conformación para los dos equipos promediar los puntos de enfrentamiento entre cada uno de los integrantes y así predecir un ganador.

De momento esta es nuestra hoja de ruta, sin embargo, estamos evaluando la posibilidad de implementar mas modelos predictivos para determinar qué composición puede ser la más favorecida ya que el dataset cuenta con una basta cantidad de información de la cual podemos intentar hacer muchas predicciones o modelos interesantes como los que se proponen en el mismo:

● ¿Podemos predecir el ganador dados los equipos?
● ¿Se puede suponer que el emparejamiento clasificado es imparcial (o ajustado para la ventaja del lado rojo)?
● ¿La región afecta significativamente las tasas de ganancias?
● ¿Podemos evaluar la información sobre los diferentes objetivos?

Se desarrolló una limpieza de los datos mediante el uso de pandas y su facilidad para manejar datos.

```
6  df.head()
```

	id	matchid	player	name	adjposition	team_role	win	kills	deaths	assists	...
0	9	10	1	Warwick	JUNGLE	1 - JUNGLE	0.0	6.0	10.0	1.0	...
1	10	10	2	Nami	DUO_SUPPORT	1 - DUO_SUPPORT	0.0	0.0	2.0	12.0	...
2	11	10	3	Draven	DUO_CARRY	1 - DUO_CARRY	0.0	7.0	8.0	5.0	...
3	12	10	4	Fiora	TOP	1 - TOP	0.0	5.0	11.0	2.0	...
4	13	10	5	Viktor	MID	1 - MID	0.0	2.0	8.0	2.0	...

5 rows × 23 columns

Imagen 1. Datos limpios

Partidas que tenían datos faltantes fueron descartadas debido a la importancia de tener datos precisos, múltiples niveles de depuración fueron utilizados.

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

	id	matchid	player	championid	ss1	ss2	role	position	win	item1	...	gameid	platformid	queueid	seasonid	duration
0	9	10	1	19	4	11	NONE	JUNGLE	0.0	3748.0	...	3187427022	EUW1	420	8	1909
1	10	10	2	267	3	4	DUO_SUPPORT	BOT	0.0	2301.0	...	3187427022	EUW1	420	8	1909
2	11	10	3	119	7	4	DUO_CARRY	BOT	0.0	1055.0	...	3187427022	EUW1	420	8	1909
3	12	10	4	114	12	4	SOLO	TOP	0.0	1029.0	...	3187427022	EUW1	420	8	1909
4	13	10	5	112	4	3	SOLO	MID	0.0	3020.0	...	3187427022	EUW1	420	8	1909
5	14	10	6	72	11	4	NONE	JUNGLE	1.0	1400.0	...	3187427022	EUW1	420	8	1909
6	15	10	7	3	4	12	SOLO	TOP	1.0	3025.0	...	3187427022	EUW1	420	8	1909
7	16	10	8	103	14	4	SOLO	MID	1.0	3135.0	...	3187427022	EUW1	420	8	1909
8	17	10	9	222	7	4	DUO_CARRY	BOT	1.0	3046.0	...	3187427022	EUW1	420	8	1909
9	18	10	10	161	14	4	DUO_SUPPORT	BOT	1.0	1058.0	...	3187427022	EUW1	420	8	1909

10 rows × 76 columns

Imagen 2. Datos faltantes

De igual manera nos fijamos en los datos relevantes que podrían ser usados para enseñarle al programa qué datos tomar, cómo clasificarlos y arrojar el resultado de quién está favorecido.

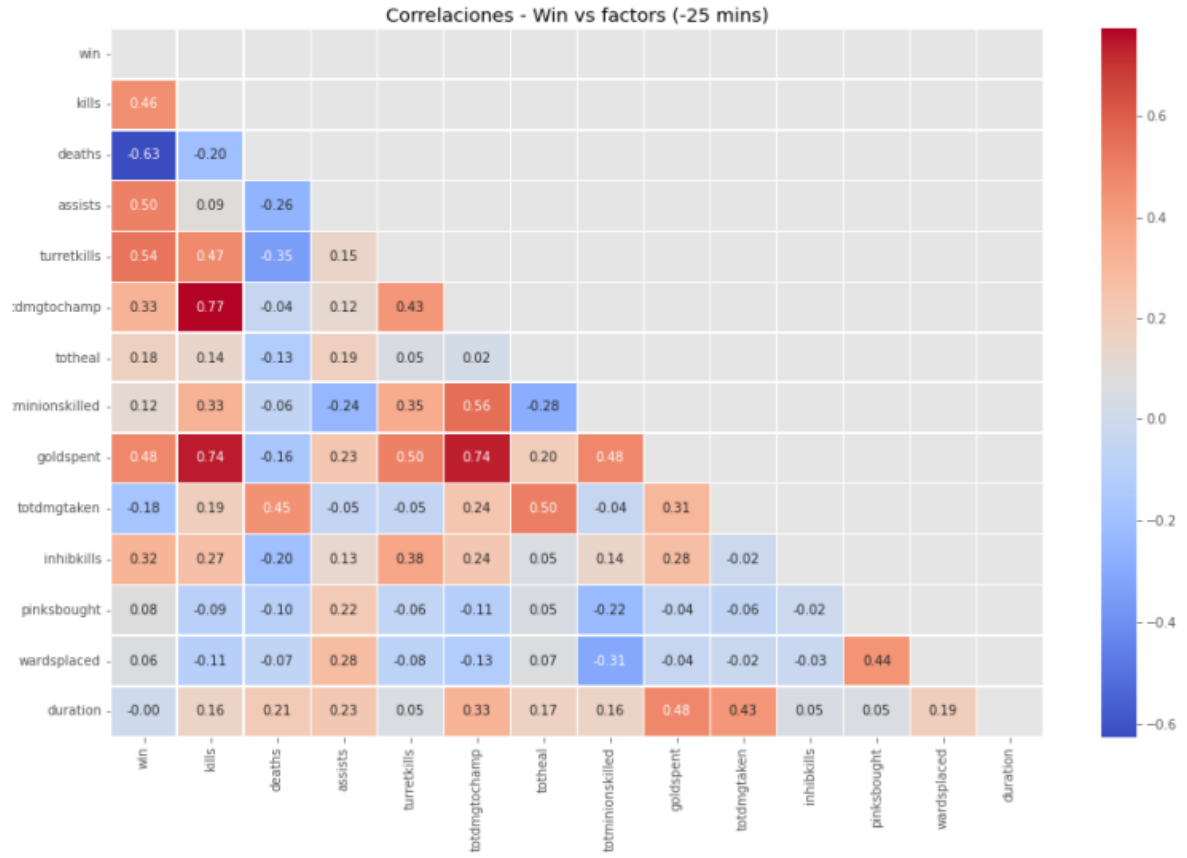

Imagen 3. Datos limpios

En la entrega dos nos enfocamos en conseguir un registro de cómo serían los enfrentamientos y en porcentajes cuál personaje estaría beneficiado estadísticamente.

```
Score dominante -/+ Significa primer, o segundo campeón dominante:

MID:
                matchup  total matches  dominant score
0          Annie vs Ryze          80.0       25.000000
1           Ahri vs Ryze         319.0       21.159875
8            Azir vs Lux         150.0      -14.666667
12  Ryze vs Twisted Fate          78.0      -14.102564
13          Ekko vs Ryze         130.0       13.846154
```

Imagen 4. Favorecimiento de enfrentamientos

Para esta entrega final nos encargamos de corregir estos enfrentamientos con los datos de x_train y además no tener valores negativos que requieren interpretación para poder utilizarlos en nuestro algoritmo de fit.

Este parámetro no era adecuado para un funcionamiento del algoritmo predictivo..

	adjposition	matchup	total matches	dominant score
0	MID	Annie vs Ryze	80	75.000000
1	MID	Ahri vs Ryze	319	71.159875
2	TOP	Gangplank vs Singed	102	65.686275
3	JUNGLE	Evelynn vs Rengar	118	65.254237
4	JUNGLE	Hecarim vs Twitch	89	65.168539

Imagen 5. Dominant score corregido para implementación

Con los datos obtenidos ahora es posible predecir quién saldría ganador en un enfrentamiento.

Lo que esperamos hacer con el conocimiento de los enfrentamientos y con los valores del x_test ahora es hallar el y_predict que nos permitirá finalmente hallar qué tan confiable es nuestro sistema.

Definimos que si el dominant score es mayor al 50% el primer jugador sería el ganador. Sin embargo si era igual o menor el ganador sería el segundo, lo que sería igual a una derrota para el primero. Es así como obtenemos nuestra predicción que contendrá solo datos 0 y 1, victoria o derrota respecto al primer personaje.

```
1  import numpy as np
2  from sklearn.metrics import accuracy_score
3
4  #y_test = y_test.values.tolist()
5
6  print(accuracy_score(y_test,y_pred))
```

```
0.5248413356955439
```

Imagen 6. Puntaje de precisión respecto a la predicción

Nuestro objetivo en primer lugar era obtener una exactitud mayor al 50% que en este caso lo hemos conseguido de una manera manual mediante un algoritmo supervisado

Algoritmo no supervisado(Tensor Flow y redes neuronales)

A Pesar de los intentos el algoritmo no supervisado por TensorFlow y Keras no funcionó correctamente ya que en el momento de la implementación de la red

neuronal esta no funciona correctamente, ya que el mayor reto en esta parte del trabajo fue trabajar con los datos categóricos que supusieron un gran problema al momento de modificarse e introducirse en la red neuronal lo que generó que al momento del entrenamiento de la misma nos generará pérdidas negativas, lo que implica una ganancia y el porcentaje de veracidad, era increíblemente bajo.

a pesar de que no funcionó creímos pertinente dejar el código en la entrega final con el objetivo de que este sea retroalimentado y así poder mejor en los aspectos que nos faltaron para que este funcionara.

```
1 # Tensorflow and Keras
2
3 df_matchup
```

	adjposition	matchup	total matches	dominant score
0	MID	Annie vs Ryze	80	75.000000
1	MID	Ahri vs Ryze	319	71.159875
2	TOP	Gangplank vs Singed	102	65.686275
3	JUNGLE	Evelynn vs Rengar	118	65.254237
4	JUNGLE	Hecarim vs Twitch	89	65.168539
...	...	...	...	...
1756	MID	Ryze vs Twisted Fate	78	35.897436
1757	JUNGLE	Graves vs Ivern	410	35.609756
1758	TOP	Shen vs Teemo	189	35.449735
1759	MID	Azir vs Lux	150	35.333333
1760	JUNGLE	Rengar vs Xin Zhao	200	32.500000

1761 rows × 4 columns

```
1 # Tratamos de organizar los datos categoricos para el procesamiento con la red neuronal
2 X = df_matchup.iloc[:,:3].values
3 X
4
```

```
array([['MID', 'Annie vs Ryze', 80],
       ['MID', 'Ahri vs Ryze', 319],
       ['TOP', 'Gangplank vs Singed', 102],
       ...,
       ['TOP', 'Shen vs Teemo', 189],
       ['MID', 'Azir vs Lux', 150],
       ['JUNGLE', 'Rengar vs Xin Zhao', 200]], dtype=object)
```

```
[86]  1 Y = df_matchup.iloc[:,3].values
      2 Y
```

```
array([75.        , 71.15987461, 65.68627451, ..., 35.44973545,
       35.33333333, 32.5       ])
```

```
[99]  1 # Organizacion de datos categoricos
      2 from sklearn.preprocessing import OneHotEncoder,LabelEncoder
      3 labelencoder_X_1 = LabelEncoder()
      4
      5 X[:,0] = labelencoder_X_1.fit_transform(X[:,0])
      6 X
```

```
array([[3, 'Annie vs Ryze', 80],
       [3, 'Ahri vs Ryze', 319],
       [4, 'Gangplank vs Singed', 102],
       ...,
       [4, 'Shen vs Teemo', 189],
       [3, 'Azir vs Lux', 150],
       [2, 'Rengar vs Xin Zhao', 200]], dtype=object)
```

```
[101]  1 onehotencoder = OneHotEncoder()
       2 X = onehotencoder.fit_transform(X).toarray()
       3 X = X[:,1:]
       4 X
       5
```

```
array([[0., 1., 0., ..., 0., 1., 0.],
       [0., 1., 0., ..., 0., 1., 0.],
       [0., 1., 0., ..., 0., 1., 0.],
       ...,
       [0., 1., 0., ..., 0., 1., 0.],
       [0., 1., 0., ..., 0., 1., 0.],
       [0., 0., 1., ..., 0., 1., 0.]])
```

[115]
```
# Creacion del X_train,X_test, Y_train, Y_test
from sklearn.model_selection import train_test_split
X_train,X_test, Y_train, Y_test = train_test_split(X,Y,test_size = 0.2, random_state = 0)
```

[116]
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
# Creacion de la red neural
from keras.engine import sequential
import keras
from keras.models import Sequential
from keras.layers import Dense

model = tf.keras.Sequential([
    tf.keras.layers.Dense(4, activation='tanh'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
model.compile(optimizer=tf.keras.optimizers.SGD(learning_rate=.5),
              loss=tf.keras.losses.BinaryCrossentropy(from_logits=False),metrics=['accuracy'])

model.fit(X_test,Y_test, epochs=100, batch_size=100)

y_predict = red.predict(X_test)
y_predict = np.round(y_predict)
y_predict
```