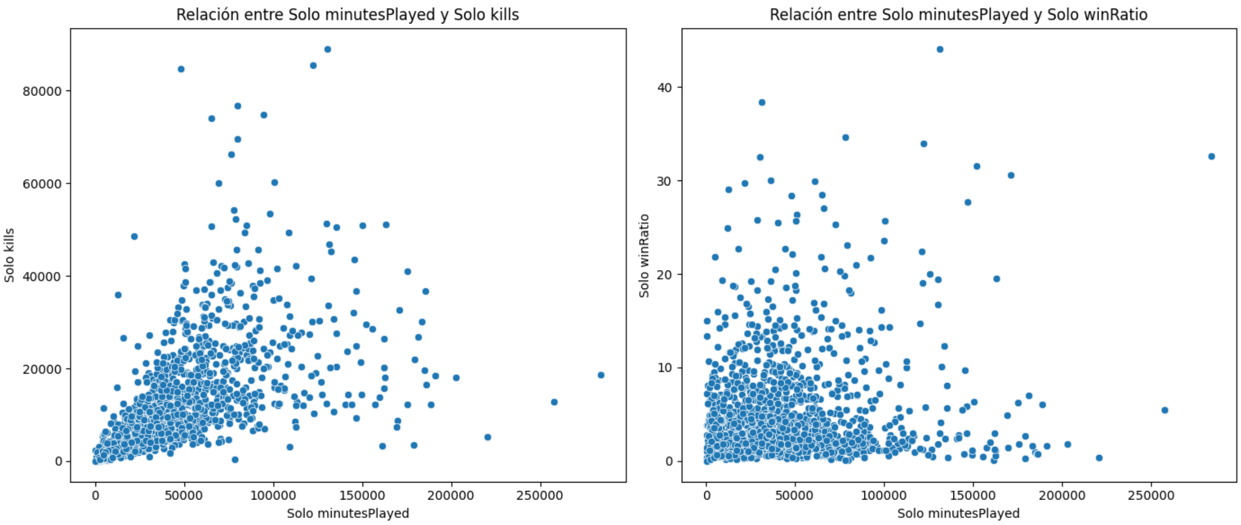
```
sb.scatterplot(x=df['Solo minutesPlayed'], y=df['Solo winRatio'])
plt.title('Relación entre Solo minutesPlayed y Solo winRatio')
plt.xlabel('Solo minutesPlayed')
plt.ylabel('Solo winRatio')

plt.tight_layout()
plt.show()
```



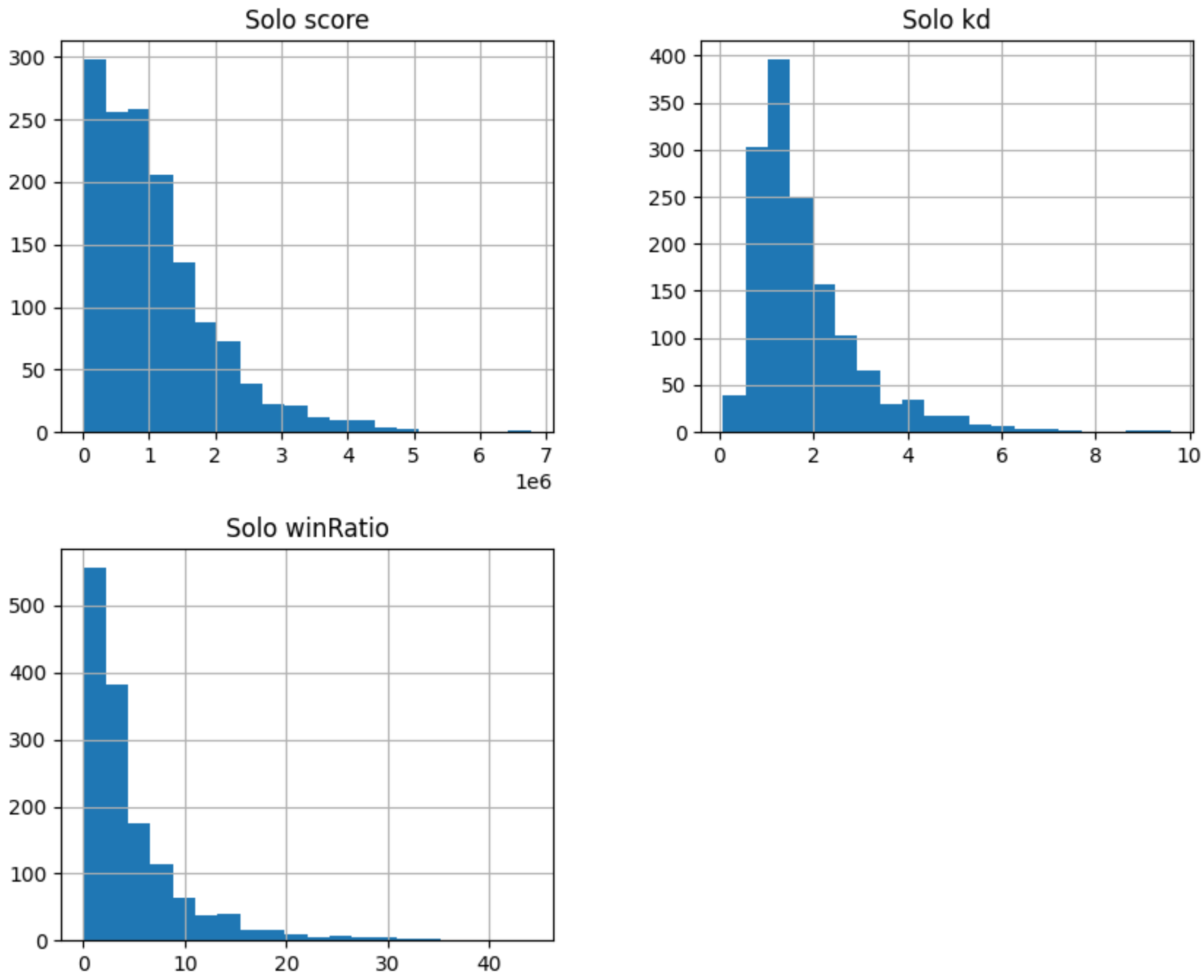
En el primer gráfico se observa que hay una tendencia positiva entre SoloMinutesPlayed y Solo Kills, según la matriz de correlación es de un 0.60, lo que indica que a medida que el tiempo de juego aumenta, también aumenta el número de eliminaciones.

Podemos decir que los jugadores que juegan más tiempo tienden a conseguir más eliminaciones. Pero, la separación de puntos nos demuestra que son muy dispersas, lo que nos señala que no todos los jugadores con mucho tiempo de juego tienen un alto número de eliminaciones.

En el gráfico de Relación entre SoloMinutesPlayed y SoloWinRatio no hay una relación clara entre estas dos variables, ya que los puntos están distribuidos de manera más dispersa y no existe una tendencia clara.

Fase 3 Data Preparation

```
# Histograma para visualizar la distribución
df[['Solo score', 'Solo kd', 'Solo winRatio']].hist(bins=20, 
figsize=(10, 8))
plt.show()
```



Solo score (Puntuación en Solitario): Este tipo de distribución sugiere que la mayor parte de los jugadores están concentrados en un rango bajo de puntuación, con un número reducido de jugadores que logran puntuaciones muy altas. Esto puede indicar una disparidad en el rendimiento, donde solo unos pocos jugadores son significativamente más exitosos que el promedio.

Solo kd (Relación K/D en Solitario): Esto indica que muchos jugadores tienden a mantener un rendimiento equilibrado (un número similar de bajas y muertes), mientras que solo algunos jugadores tienen un rendimiento mucho más alto (con un kd por encima de 2 o 3). Los valores extremos pueden ser jugadores con mucha habilidad o que se enfrentan a oponentes de menor nivel.

Solo winRatio (Porcentaje de Victorias en Solitario): La mayoría de los jugadores probablemente tengan dificultades para ganar en partidas en solitario. Solo un pequeño porcentaje logra un winRatio alto, lo cual es esperable, ya que las victorias son un resultado más exclusivo en un entorno competitivo como Fortnite.

Todas las variables muestran una distribución sesgada a la derecha, lo que sugiere que la mayoría de los jugadores tienen un rendimiento moderado, mientras que solo unos pocos logran resultados significativamente mejores.

Para mejorar el rendimiento del modelo vamos a normalizar los datos así obtener un mejor nivel predictivo de los datos.

Primero seleccionamos las columnas que vamos a normalizar y utilizamos MinMaxScaler para escalar a valores entre 0 y 1.

```
columnasNormalizar = ['Solo score', 'Solo top1', 'Solo kd', 'Solo
winRatio', 'Solo matches', 'Solo kills', 'Solo minutesPlayed']
```

Columnas antes de la normalización

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

```
{"column_count":8,"columns":
[{"dtype":"object","name":"Player","stats":{"categories":
[{"count":1,"name":"ᴮᵒˢˢ ン"},{"count":1,"name":"Raпger"},
{"count":8,"name":"8 others"}],"nan_count":0,"unique_count":10}},
{"dtype":"int64","name":"Solo score","stats":{"histogram":
[{"bin_end":691496.4,"bin_start":15872,"count":3},
{"bin_end":1367120.8,"bin_start":691496.4,"count":2},
{"bin_end":2042745.2000000002,"bin_start":1367120.8,"count":0},
{"bin_end":2718369.6,"bin_start":2042745.2000000002,"count":2},
{"bin_end":3393994,"bin_start":2718369.6,"count":1},
{"bin_end":4069618.4000000004,"bin_start":3393994,"count":0},
{"bin_end":4745242.8,"bin_start":4069618.4000000004,"count":1},
{"bin_end":5420867.2,"bin_start":4745242.8,"count":0},
{"bin_end":6096491.600000001,"bin_start":5420867.2,"count":0},
{"bin_end":6772116,"bin_start":6096491.600000001,"count":1}],"max":"67
72116","min":"15872","nan_count":0,"unique_count":10}},
{"dtype":"int64","name":"Solo top1","stats":{"histogram":
[{"bin_end":653.4,"bin_start":6,"count":3},
{"bin_end":1300.8,"bin_start":653.4,"count":0},
{"bin_end":1948.1999999999998,"bin_start":1300.8,"count":5},
{"bin_end":2595.6,"bin_start":1948.1999999999998,"count":0},
{"bin_end":3243,"bin_start":2595.6,"count":0},
{"bin_end":3890.3999999999996,"bin_start":3243,"count":0},
{"bin_end":4537.8,"bin_start":3890.3999999999996,"count":0},
{"bin_end":5185.2,"bin_start":4537.8,"count":1},
{"bin_end":5832.599999999999,"bin_start":5185.2,"count":0},
{"bin_end":6480,"bin_start":5832.599999999999,"count":1}],"max":"6480"
,"min":"6","nan_count":0,"unique_count":10}},
{"dtype":"float64","name":"Solo kd","stats":{"histogram":
[{"bin_end":1.383,"bin_start":0.47,"count":3},
{"bin_end":2.296,"bin_start":1.383,"count":1},
{"bin_end":3.2089999999999996,"bin_start":2.296,"count":0},
{"bin_end":4.122,"bin_start":3.2089999999999996,"count":2},
{"bin_end":5.034999999999999,"bin_start":4.122,"count":1},
{"bin_end":5.9479999999999995,"bin_start":5.034999999999999,"count":0}
,
{"bin_end":6.860999999999999,"bin_start":5.9479999999999995,"count":1}
```

```
,
{"bin_end":7.773999999999999,"bin_start":6.860999999999999,"count":0},
{"bin_end":8.687,"bin_start":7.773999999999999,"count":0},
{"bin_end":9.6,"bin_start":8.687,"count":2}],"max":"9.6","min":"0.47",
"nan_count":0,"unique_count":10}},{"dtype":"float64","name":"Solo
winRatio","stats":{"histogram":
[{"bin_end":3.85,"bin_start":0.5,"count":3},
{"bin_end":7.2,"bin_start":3.85,"count":0},
{"bin_end":10.55,"bin_start":7.2,"count":2},
{"bin_end":13.9,"bin_start":10.55,"count":0},
{"bin_end":17.25,"bin_start":13.9,"count":0},
{"bin_end":20.6,"bin_start":17.25,"count":1},
{"bin_end":23.95,"bin_start":20.6,"count":0},
{"bin_end":27.3,"bin_start":23.95,"count":0},
{"bin_end":30.650000000000002,"bin_start":27.3,"count":2},
{"bin_end":34,"bin_start":30.650000000000002,"count":2}],"max":"34.0",
"min":"0.5","nan_count":0,"unique_count":10}},
{"dtype":"int64","name":"Solo matches","stats":{"histogram":
[{"bin_end":2372.5,"bin_start":429,"count":2},
{"bin_end":4316,"bin_start":2372.5,"count":1},
{"bin_end":6259.5,"bin_start":4316,"count":2},
{"bin_end":8203,"bin_start":6259.5,"count":0},
{"bin_end":10146.5,"bin_start":8203,"count":0},
{"bin_end":12090,"bin_start":10146.5,"count":1},
{"bin_end":14033.5,"bin_start":12090,"count":2},
{"bin_end":15977,"bin_start":14033.5,"count":0},
{"bin_end":17920.5,"bin_start":15977,"count":0},
{"bin_end":19864,"bin_start":17920.5,"count":2}],"max":"19864","min":"
429","nan_count":0,"unique_count":10}},{"dtype":"int64","name":"Solo
kills","stats":{"histogram":
[{"bin_end":8728.1,"bin_start":200,"count":3},
{"bin_end":17256.2,"bin_start":8728.1,"count":0},
{"bin_end":25784.300000000003,"bin_start":17256.2,"count":2},
{"bin_end":34312.4,"bin_start":25784.300000000003,"count":0},
{"bin_end":42840.5,"bin_start":34312.4,"count":3},
{"bin_end":51368.600000000006,"bin_start":42840.5,"count":0},
{"bin_end":59896.700000000004,"bin_start":51368.600000000006,"count":0
},{"bin_end":68424.8,"bin_start":59896.700000000004,"count":1},
{"bin_end":76952.90000000001,"bin_start":68424.8,"count":0},
{"bin_end":85481,"bin_start":76952.90000000001,"count":1}],"max":"8548
1","min":"200","nan_count":0,"unique_count":10}},
{"dtype":"int64","name":"Solo minutesPlayed","stats":{"histogram":
[{"bin_end":29048.6,"bin_start":739,"count":3},
{"bin_end":57358.2,"bin_start":29048.6,"count":2},
{"bin_end":85667.79999999999,"bin_start":57358.2,"count":2},
{"bin_end":113977.4,"bin_start":85667.79999999999,"count":1},
{"bin_end":142287,"bin_start":113977.4,"count":1},
{"bin_end":170596.59999999998,"bin_start":142287,"count":0},
{"bin_end":198906.19999999998,"bin_start":170596.59999999998,"count":0
},{"bin_end":227215.8,"bin_start":198906.19999999998,"count":0},
{"bin_end":255525.4,"bin_start":227215.8,"count":0},
{"bin_end":283835,"bin_start":255525.4,"count":1}],"max":"283835","min
":"739","nan_count":0,"unique_count":10}},
{"dtype":"int64","name":"_deepnote_index_column"}],"row_count":10,"row
s":[{"Player":"ᴮᵒˢˢン","Solo kd":1.39,"Solo kills":18610,"Solo
matches":19864,"Solo minutesPlayed":283835,"Solo score":6772116,"Solo
top1":6480,"Solo winRatio":32.6,"_deepnote_index_column":8},
{"Player":"Raпger","Solo kd":9.6,"Solo kills":85481,"Solo
matches":13488,"Solo minutesPlayed":122171,"Solo score":4519465,"Solo
top1":4582,"Solo winRatio":34,"_deepnote_index_column":2},
{"Player":"Twitch Kayotica","Solo kd":3.23,"Solo kills":39131,"Solo
matches":13438,"Solo minutesPlayed":96777,"Solo score":2919037,"Solo
top1":1310,"Solo winRatio":9.7,"_deepnote_index_column":5},
{"Player":"Prospering","Solo kd":4.37,"Solo kills":36328,"Solo
matches":10150,"Solo minutesPlayed":81389,"Solo score":2476763,"Solo
top1":1828,"Solo winRatio":18,"_deepnote_index_column":0},
{"Player":"FaZe Replays","Solo kd":3.84,"Solo kills":66161,"Solo
matches":18670,"Solo minutesPlayed":76258,"Solo score":2389537,"Solo
top1":1454,"Solo winRatio":7.8,"_deepnote_index_column":6},
{"Player":"Twitch.GryphonRB","Solo kd":6.32,"Solo kills":19591,"Solo
matches":4429,"Solo minutesPlayed":36245,"Solo score":1136282,"Solo
top1":1327,"Solo winRatio":30,"_deepnote_index_column":4},
{"Player":"Idk_Pi","Solo kd":0.84,"Solo kills":3005,"Solo
matches":3687,"Solo minutesPlayed":32453,"Solo score":752869,"Solo
top1":121,"Solo winRatio":3.3,"_deepnote_index_column":3},
{"Player":"BH nixxxay","Solo kd":8.71,"Solo kills":35895,"Solo
matches":5817,"Solo minutesPlayed":12732,"Solo score":439562,"Solo
top1":1694,"Solo winRatio":29.1,"_deepnote_index_column":1},
{"Player":"CIUPEA 144.HZ","Solo kd":0.61,"Solo kills":1174,"Solo
matches":1938,"Solo minutesPlayed":2441,"Solo score":54479,"Solo
top1":9,"Solo winRatio":0.5,"_deepnote_index_column":9},
{"Player":"NiteGamerYT 190k","Solo kd":0.47,"Solo kills":200,"Solo
matches":429,"Solo minutesPlayed":739,"Solo score":15872,"Solo
top1":6,"Solo winRatio":1.4,"_deepnote_index_column":7}]}
```

Columnas después de la normalización

```python
# Aplicar Min-Max Scaler
scaler = MinMaxScaler()
dfNormalizado = df.copy()  # Crear una copia del DataFrame
dfNormalizado[columnasNormalizar] =
scaler.fit_transform(df[columnasNormalizar])
dfNormalizado.head(10)
```

```
{"column_count":8,"columns":
[{"dtype":"object","name":"Player","stats":{"categories":
[{"count":1,"name":"ᴮᵒˢˢン"},{"count":1,"name":"Raпger"},
{"count":8,"name":"8 others"}],"nan_count":0,"unique_count":10}},
{"dtype":"float64","name":"Solo score","stats":{"histogram":
[{"bin_end":0.10151855965641314,"bin_start":1.6872885071257243e-
3,"count":3},
{"bin_end":0.20134983080570054,"bin_start":0.10151855965641314,"count"
:2},
{"bin_end":0.301181101954988,"bin_start":0.20134983080570054,"count":0
},
{"bin_end":0.4010123731042754,"bin_start":0.301181101954988,"count":2}
,
{"bin_end":0.5008436442535628,"bin_start":0.4010123731042754,"count":1
},
{"bin_end":0.6006749154028502,"bin_start":0.5008436442535628,"count":0
},
{"bin_end":0.7005061865521376,"bin_start":0.6006749154028502,"count":1
},
{"bin_end":0.8003374577014251,"bin_start":0.7005061865521376,"count":0
},
{"bin_end":0.9001687288507125,"bin_start":0.8003374577014251,"count":0
},
{"bin_end":0.9999999999999999,"bin_start":0.9001687288507125,"count":1
}],"max":"0.9999999999999999","min":"0.0016872885071257243","nan_count
":0,"unique_count":10}},{"dtype":"float64","name":"Solo top1","stats":
{"histogram":
[{"bin_end":0.10083333333333334,"bin_start":9.25925925925926e-
4,"count":3},
{"bin_end":0.20074074074074075,"bin_start":0.10083333333333334,"count"
:0},
{"bin_end":0.30064814814814816,"bin_start":0.20074074074074075,"count"
:5},
{"bin_end":0.4005555555555556,"bin_start":0.30064814814814816,"count":
0},
{"bin_end":0.500462962962963,"bin_start":0.4005555555555556,"count":0}
,
{"bin_end":0.6003703703703703,"bin_start":0.500462962962963,"count":0}
,
{"bin_end":0.7002777777777778,"bin_start":0.6003703703703703,"count":0
},
{"bin_end":0.8001851851851852,"bin_start":0.7002777777777778,"count":1
},
{"bin_end":0.9000925925925927,"bin_start":0.8001851851851852,"count":0
},
{"bin_end":1,"bin_start":0.9000925925925927,"count":1}],"max":"1.0","m
in":"0.000925925925925926","nan_count":0,"unique_count":10}},
{"dtype":"float64","name":"Solo kd","stats":{"histogram":
[{"bin_end":0.13686974789915968,"bin_start":4.096638655462185e-
2,"count":3},
{"bin_end":0.23277310924369748,"bin_start":0.13686974789915968,"count"
:1},
{"bin_end":0.32867647058823535,"bin_start":0.23277310924369748,"count"
:0},
{"bin_end":0.42457983193277316,"bin_start":0.32867647058823535,"count"
:2},
{"bin_end":0.520483193277311,"bin_start":0.42457983193277316,"count":1
},
{"bin_end":0.6163865546218488,"bin_start":0.520483193277311,"count":0}
,
{"bin_end":0.7122899159663866,"bin_start":0.6163865546218488,"count":1
},
{"bin_end":0.8081932773109244,"bin_start":0.7122899159663866,"count":0
},
{"bin_end":0.9040966386554622,"bin_start":0.8081932773109244,"count":0
},
{"bin_end":1,"bin_start":0.9040966386554622,"count":2}],"max":"1.0","m
in":"0.04096638655462185","nan_count":0,"unique_count":10}},
{"dtype":"float64","name":"Solo winRatio","stats":{"histogram":
[{"bin_end":8.730158730158731e-2,"bin_start":1.1337868480725623e-
2,"count":3},
{"bin_end":0.163265306122449,"bin_start":8.730158730158731e-
2,"count":0},
{"bin_end":0.2392290249433107,"bin_start":0.163265306122449,"count":2}
,
{"bin_end":0.31519274376417233,"bin_start":0.2392290249433107,"count":
0},
{"bin_end":0.391156462585034,"bin_start":0.31519274376417233,"count":0
},
{"bin_end":0.46712018140589573,"bin_start":0.391156462585034,"count":1
},
{"bin_end":0.5430839002267573,"bin_start":0.46712018140589573,"count":
0},
{"bin_end":0.6190476190476191,"bin_start":0.5430839002267573,"count":0
},
{"bin_end":0.6950113378684808,"bin_start":0.6190476190476191,"count":2
},
{"bin_end":0.7709750566893424,"bin_start":0.6950113378684808,"count":2
}],"max":"0.7709750566893424","min":"0.011337868480725623","nan_count"
:0,"unique_count":10}},{"dtype":"float64","name":"Solo
matches","stats":{"histogram":[{"bin_end":6.425377842389664e-
2,"bin_start":1.0944400252352085e-2,"count":2},
{"bin_end":0.1175631565954412,"bin_start":6.425377842389664e-
2,"count":1},
{"bin_end":0.17087253476698575,"bin_start":0.1175631565954412,"count":
2},
{"bin_end":0.2241819129385303,"bin_start":0.17087253476698575,"count":
0},
{"bin_end":0.27749129111007487,"bin_start":0.2241819129385303,"count":
0},
{"bin_end":0.33080066928161944,"bin_start":0.27749129111007487,"count"
:1},
{"bin_end":0.38411004745316396,"bin_start":0.33080066928161944,"count"
:2},
{"bin_end":0.43741942562470854,"bin_start":0.38411004745316396,"count"
:0},
{"bin_end":0.4907288037962531,"bin_start":0.43741942562470854,"count":
0},
{"bin_end":0.5440381819677976,"bin_start":0.4907288037962531,"count":2
}],"max":"0.5440381819677976","min":"0.010944400252352085","nan_count"
:0,"unique_count":10}},{"dtype":"float64","name":"Solo kills","stats":
{"histogram":[{"bin_end":9.775942468678016e-
2,"bin_start":1.9326928479128042e-3,"count":3},
{"bin_end":0.1935861565256475,"bin_start":9.775942468678016e-
2,"count":0},
{"bin_end":0.28941288836451484,"bin_start":0.1935861565256475,"count":
2},
{"bin_end":0.3852396202033822,"bin_start":0.28941288836451484,"count":
0},
{"bin_end":0.48106635204224957,"bin_start":0.3852396202033822,"count":
3},
{"bin_end":0.5768930838811168,"bin_start":0.48106635204224957,"count":
0},
{"bin_end":0.6727198157199842,"bin_start":0.5768930838811168,"count":0
},
{"bin_end":0.7685465475588515,"bin_start":0.6727198157199842,"count":1
},
{"bin_end":0.8643732793977189,"bin_start":0.7685465475588515,"count":0
},
{"bin_end":0.9602000112365863,"bin_start":0.8643732793977189,"count":1
}],"max":"0.9602000112365863","min":"0.0019326928479128042","nan_count
":0,"unique_count":10}},{"dtype":"float64","name":"Solo
minutesPlayed","stats":{"histogram":
[{"bin_end":0.10167862494490965,"bin_start":1.8651388276773907e-
3,"count":3},
{"bin_end":0.2014921110621419,"bin_start":0.10167862494490965,"count":
2},
{"bin_end":0.30130559717937416,"bin_start":0.2014921110621419,"count":
2},
{"bin_end":0.40111908329660645,"bin_start":0.30130559717937416,"count"
:1},
{"bin_end":0.5009325694138387,"bin_start":0.40111908329660645,"count":
1},
{"bin_end":0.6007460555310709,"bin_start":0.5009325694138387,"count":0
},
{"bin_end":0.7005595416483031,"bin_start":0.6007460555310709,"count":0
},
{"bin_end":0.8003730277655354,"bin_start":0.7005595416483031,"count":0
},
{"bin_end":0.9001865138827677,"bin_start":0.8003730277655354,"count":0
},
{"bin_end":1,"bin_start":0.9001865138827677,"count":1}],"max":"1.0","m
in":"0.0018651388276773907","nan_count":0,"unique_count":10}},
{"dtype":"int64","name":"_deepnote_index_column"}],"row_count":10,"row
s":[{"Player":"ᴮᵒˢˢン","Solo kd":0.1376050420168067,"Solo
kills":0.20879824709253328,"Solo matches":0.5440381819677976,"Solo
minutesPlayed":1,"Solo score":0.9999999999999999,"Solo top1":1,"Solo
winRatio":0.7392290249433107,"_deepnote_index_column":8},
{"Player":"Raпger","Solo kd":1,"Solo kills":0.9602000112365863,"Solo
matches":0.3691472145267027,"Solo
minutesPlayed":0.4300079330101366,"Solo
score":0.6671449213709371,"Solo top1":0.7070987654320987,"Solo
winRatio":0.7709750566893424,"_deepnote_index_column":2},
{"Player":"Twitch Kayotica","Solo kd":0.33088235294117646,"Solo
kills":0.4393842350693859,"Solo matches":0.3677757357983378,"Solo
minutesPlayed":0.340474217717056,"Solo
score":0.43066328805083826,"Solo top1":0.2021604938271605,"Solo
winRatio":0.21995464852607707,"_deepnote_index_column":5},
{"Player":"Prospering","Solo kd":0.45063025210084034,"Solo
kills":0.4078880836002023,"Solo matches":0.2775872946210604,"Solo
minutesPlayed":0.2862194799471133,"Solo
score":0.3653122077695064,"Solo top1":0.2820987654320988,"Solo
winRatio":0.40816326530612246,"_deepnote_index_column":0},
{"Player":"FaZe Replays","Solo kd":0.3949579831932773,"Solo
kills":0.7431091634361481,"Solo matches":0.5112872699344433,"Solo
minutesPlayed":0.2681286910533275,"Solo
score":0.3524235766467686,"Solo top1":0.2243827160493827,"Solo
winRatio":0.17687074829931973,"_deepnote_index_column":6},
{"Player":"Twitch.GryphonRB","Solo kd":0.6554621848739497,"Solo
kills":0.2198213382774313,"Solo matches":0.12066269852154593,"Solo
minutesPlayed":0.12705156456588806,"Solo
score":0.16724074470020153,"Solo top1":0.20478395061728394,"Solo
winRatio":0.6802721088435374,"_deepnote_index_column":4},
{"Player":"Idk_Pi","Solo kd":7.983193277310924e-2,"Solo
kills":3.345131748974662e-2,"Solo matches":0.10030995419261048,"Solo
minutesPlayed":0.1136817981489643,"Solo
score":0.11058706676145073,"Solo top1":1.867283950617284e-2,"Solo
winRatio":7.482993197278912e-2,"_deepnote_index_column":3},
{"Player":"BH nixxxay","Solo kd":0.9065126050420169,"Solo
kills":0.40302264172144503,"Solo matches":0.1587349480209562,"Solo
minutesPlayed":4.4149845747025115e-2,"Solo score":6.429235616489769e-
2,"Solo top1":0.26141975308641974,"Solo
winRatio":0.6598639455782314,"_deepnote_index_column":1},
{"Player":"CIUPEA 144.HZ","Solo kd":5.567226890756302e-2,"Solo
kills":1.287712792853531e-2,"Solo matches":5.2335628274405466e-2,"Solo
minutesPlayed":7.866020273248127e-3,"Solo score":7.391916530122732e-
3,"Solo top1":1.3888888888888889e-3,"Solo
winRatio":1.1337868480725623e-2,"_deepnote_index_column":9},
{"Player":"NiteGamerYT 190k","Solo kd":4.096638655462185e-2,"Solo
kills":1.9326928479128042e-3,"Solo matches":1.0944400252352085e-
```

```
2,"Solo minutesPlayed":1.8651388276773907e-3,"Solo
score":1.6872885071257243e-3,"Solo top1":9.25925925925926e-4,"Solo
winRatio":3.1746031746031744e-2,"_deepnote_index_column":7}]}
```

aValor cercano a 0: El jugador tiene un bajo rendimiento en comparación con otros jugadores para esa métrica específica. Valor cercano a 1: El jugador tiene un alto rendimiento en comparación con otros jugadores para esa métrica específica.

"Boss" es el mejor jugador en términos de Solo score, Solo top1, Solo kd y Solo minutesPlayed, porque sus valores están en 1, lo que indica que está al tope de las métricas de rendimiento en comparación con los demás jugadores.

La normalización nos ayuda a estandarizar las escalas de las diferentes variables para que los algoritmos de machine learning puedan interpretar correctamente los datos, evitando que variables con mayor rango afecten de manera desproporcionada los resultados.

Tratamiento de outliers

```
# Calcular el IQR
Q1 = df['Solo kd'].quantile(0.25)
Q3 = df['Solo kd'].quantile(0.75)
IQR = Q3 - Q1

# Definir los límites inferior y superior
lim_inferior = Q1 - 1.5 * IQR
lim_superior = Q3 + 1.5 * IQR

# Filtrar datos dentro de los límites
df_solo_sin_outliers = df[(df['Solo kd'] >= lim_inferior) & (df['Solo
kd'] <= lim_superior)]

print("Número de filas antes de eliminar outliers:", len(df))
print("Número de filas después de eliminar outliers:",
len(df_solo_sin_outliers))
```

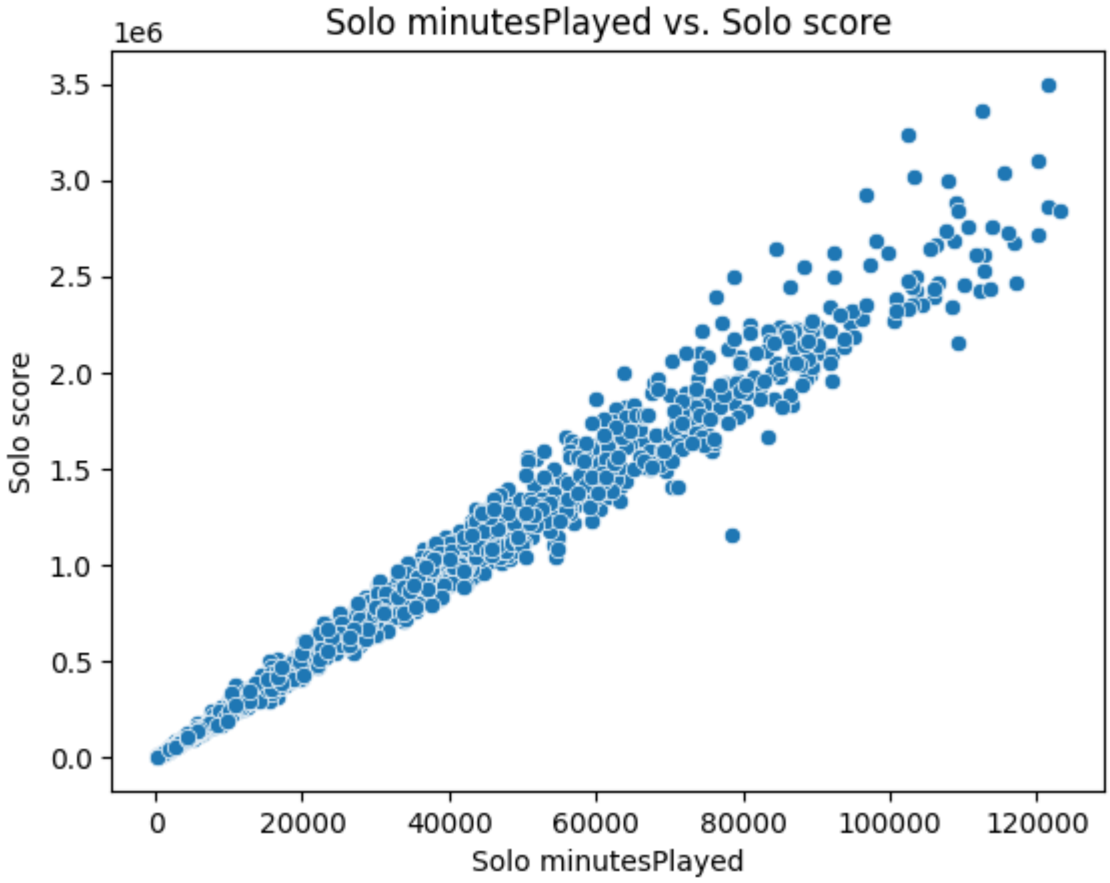
```
Número de filas antes de eliminar outliers: 1435
Número de filas después de eliminar outliers: 1343
```

```
# Verificar límites del IQR
print(f"Límite inferior: {lim_inferior}, Límite superior:
{lim_superior}")
```

```
Límite inferior: -0.675, Límite superior: 3.9250000000000003
```

```
# Cálculo del IQR para Solo minutesPlayed
Q1_mp = df['Solo minutesPlayed'].quantile(0.25)
Q3_mp = df['Solo minutesPlayed'].quantile(0.75)
IQR_mp = Q3_mp - Q1_mp
```

```
# Límite inferior y superior para Solo minutesPlayed
lim_inferior_mp = Q1_mp - 1.5 * IQR_mp
lim_superior_mp = Q3_mp + 1.5 * IQR_mp

# Filtrar datos sin outliers en Solo kd y Solo minutesPlayed
df_solo_sin_outliers =
df_solo_sin_outliers[(df_solo_sin_outliers['Solo minutesPlayed'] >=
lim_inferior_mp) &

(df_solo_sin_outliers['Solo minutesPlayed'] <= lim_superior_mp)]

# Visualización final del scatterplot sin outliers
sb.scatterplot(x='Solo minutesPlayed', y='Solo score',
data=df_solo_sin_outliers)
plt.title('Solo minutesPlayed vs. Solo score')
plt.show()
```



El gráfico resultante muestra una relación positiva entre el tiempo jugado en modo solo (Solo minutesPlayed) y la puntuación obtenida en ese modo (Solo score). Es decir, a medida que los jugadores pasan más tiempo jugando, tienden a obtener puntuaciones más altas.

Esta relación es consistente y sigue una tendencia lineal, aunque se observan algunas dispersiones (pequeñas variaciones) a medida que aumenta el número de minutos jugados. Al haber filtrado los outliers, se han eliminado valores atípicos que podrían haber distorsionado la interpretación de la tendencia general. El gráfico se concentra en los datos que representan el comportamiento típico de los jugadores.

Tratamiento de outliers para conocer la existencia de relación significativa entre el tiempo jugado (Solo minutesPlayed) y el ratio de victorias (Solo winRatio)?

```
# Cálculo del IQR para Solo minutesPlayed
Q1_minutes = df['Solo minutesPlayed'].quantile(0.25)
Q3_minutes = df['Solo minutesPlayed'].quantile(0.75)
IQR_minutes = Q3_minutes - Q1_minutes

# Cálculo del IQR para Solo kills
Q1_kills = df['Solo kills'].quantile(0.25)
Q3_kills = df['Solo kills'].quantile(0.75)
IQR_kills = Q3_kills - Q1_kills

# Definir límites para Solo minutesPlayed
lower_bound_minutes = Q1_minutes - 1.5 * IQR_minutes
upper_bound_minutes = Q3_minutes + 1.5 * IQR_minutes

# Definir límites para Solo kills
lower_bound_kills = Q1_kills - 1.5 * IQR_kills
upper_bound_kills = Q3_kills + 1.5 * IQR_kills

# Filtrar datos para eliminar outliers
df_sin_outliers = df[(df['Solo minutesPlayed'] >= lower_bound_minutes)
& (df['Solo minutesPlayed'] <= upper_bound_minutes) &
                     (df['Solo kills'] >= lower_bound_kills) &
(df['Solo kills'] <= upper_bound_kills)]

# Mostrar los datos sin outliers
print(df_sin_outliers)
```

```
                Player  Solo score  Solo top1  Solo kd  Solo winRatio
\
3               Idk_Pi      752869        121     0.84            3.3

4      Twitch.GryphonRB     1136282       1327     6.32           30.0

7      NiteGamerYT 190k       15872          6     0.47            1.4

9         CIUPEA 144.HZ       54479          9     0.61            0.5

10              曼巴精神の R6      208411        103     4.17
12.2
...                 ...         ...        ...      ...            ...
```

```
1430          im bloom      106294         26     2.03            3.5

1431     Twitch kaOzs_      803918        396     3.42           13.2

1432           slxyher      603939        310     1.69            5.1

1433         RD-Antony      747295         71     1.10            1.9

1434       O-HO-HO-HO!     1378689        161     1.68            2.3


      Solo matches  Solo kills  Solo minutesPlayed
3             3687        3005               32453
4             4429       19591               36245
7              429         200                 739
9             1938        1174                2441
10             844        3093                6661
...            ...         ...                 ...
1430           736        1440                4470
1431          3001        8911               27375
1432          6035        9679               20545
1433          3763        4048               31147
1434          7105       11661               57572

[1322 rows x 8 columns]
```

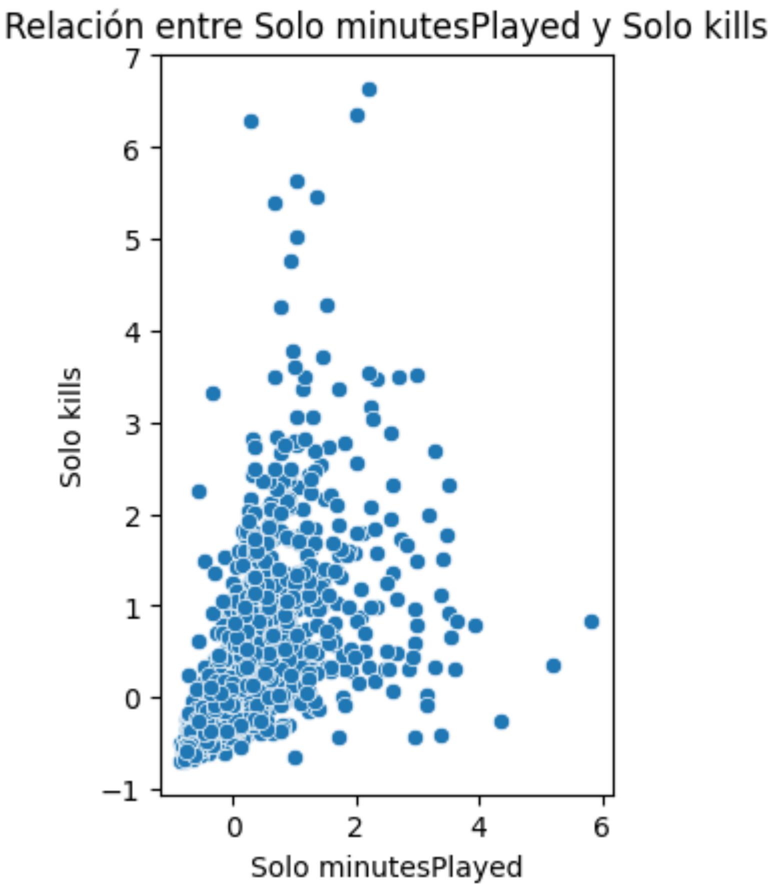
Usamos RobustScaler de scikit-learn, para reducir impacto de los outliers escalando los datos de acuerdo al IQR

```
scaler = RobustScaler()
df[['Solo minutesPlayed', 'Solo kills']] =
scaler.fit_transform(df[['Solo minutesPlayed', 'Solo kills']])
```

```
/tmp/ipykernel_670/3731571679.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df[['Solo minutesPlayed', 'Solo kills']] =
scaler.fit_transform(df[['Solo minutesPlayed', 'Solo kills']])
```

```
# Visualizar outliers
plt.subplot(1, 2, 1)
sb.scatterplot(x=df['Solo minutesPlayed'], y=df['Solo kills'])
plt.title('Relación entre Solo minutesPlayed y Solo kills')
plt.xlabel('Solo minutesPlayed')
plt.ylabel('Solo kills')
```

```
Text(0, 0.5, 'Solo kills')
```



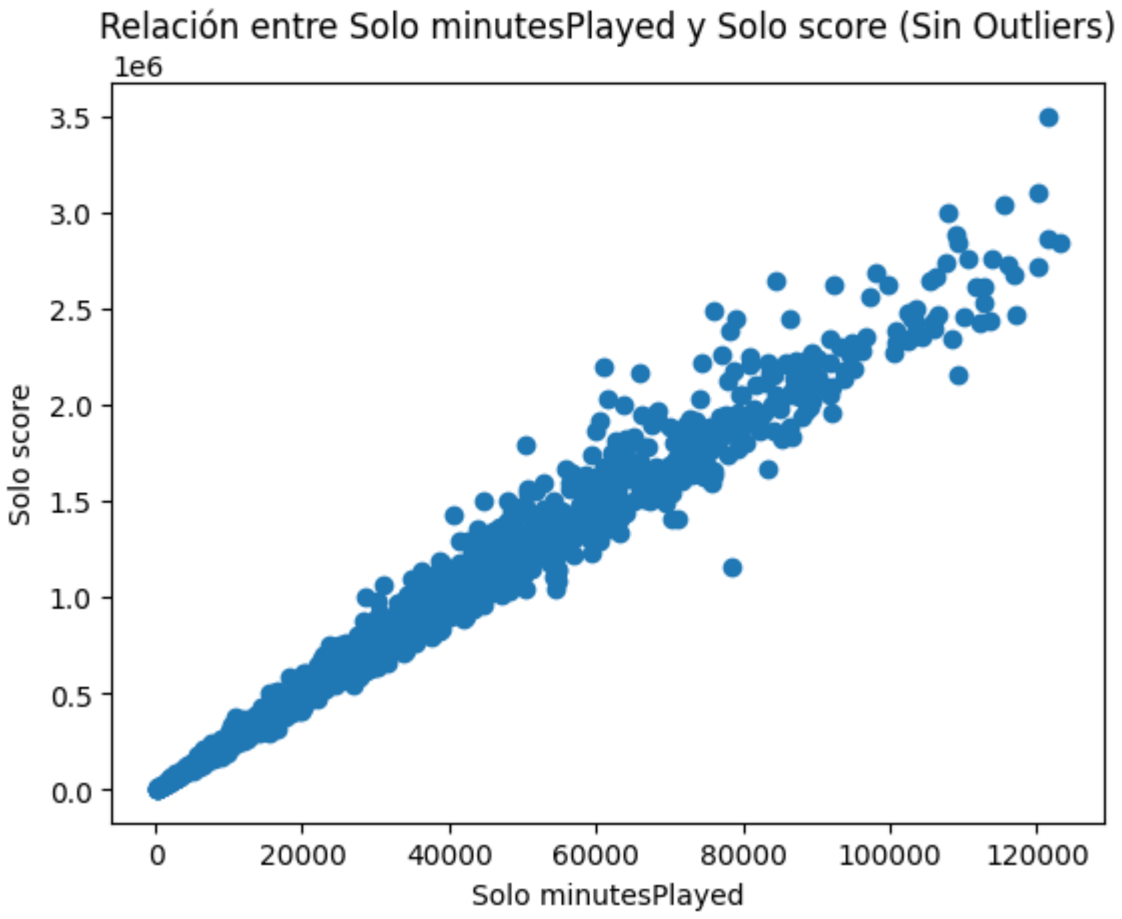
Tratamiento de outliers para conocer qué factores influyen en la puntuación (Solo score) de un jugador

```
Q1 = df['Solo score'].quantile(0.25)
Q3 = df['Solo score'].quantile(0.75)
IQR = Q3 - Q1

# Filtra los datos que no son outliers
dfSinOutliers = df[~((df['Solo score'] < (Q1 - 1.5 * IQR)) | (df['Solo
score'] > (Q3 + 1.5 * IQR)))]

# Mostrar los datos sin outliers
print(dfSinOutliers)
```

```
                Player  Solo score  Solo top1  Solo kd  Solo winRatio
\
0           Prospering     2476763       1828     4.37           18.0

1           BH nixxxay      439562       1694     8.71           29.1
```

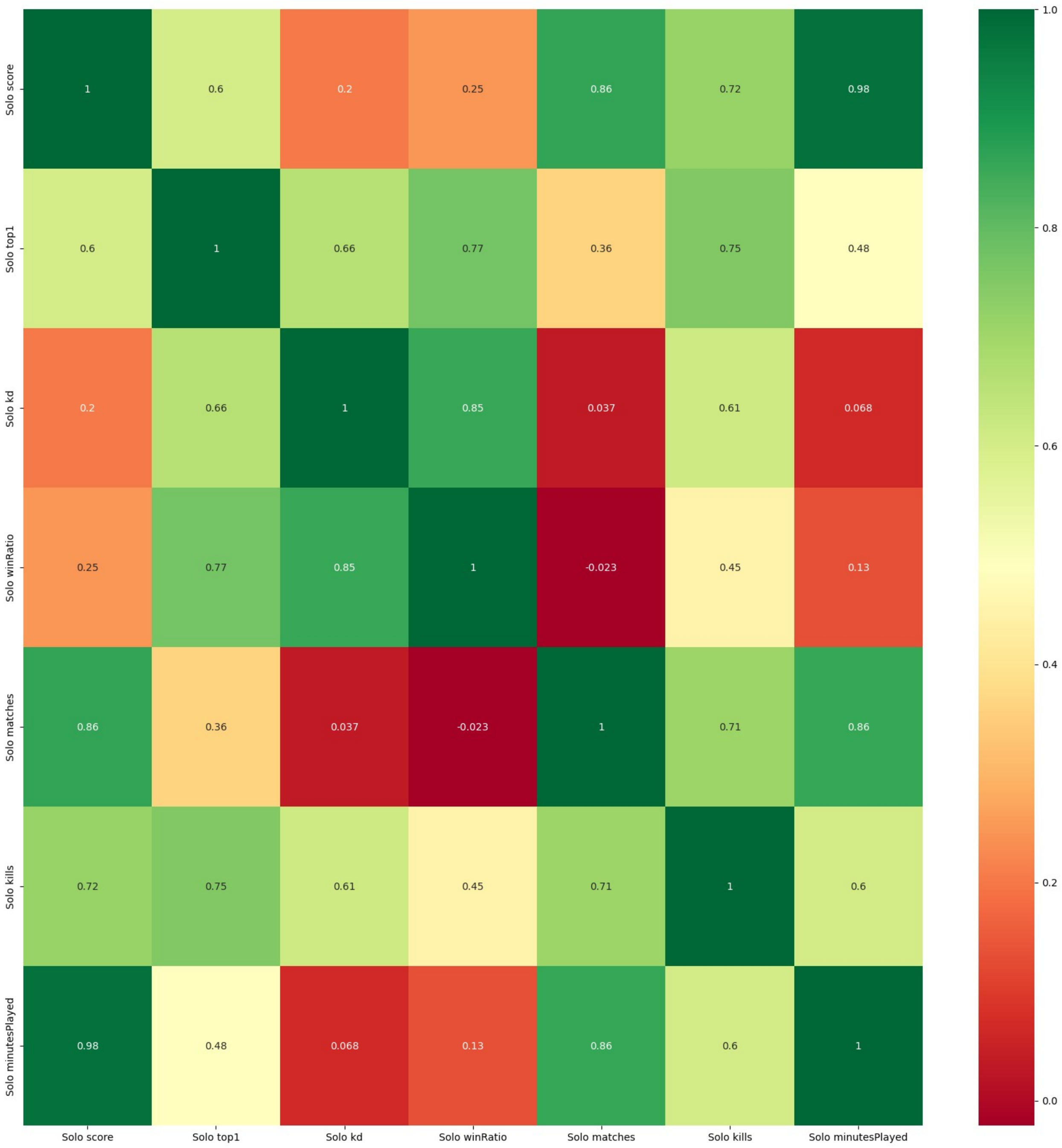
```
3               Idk_Pi      752869        121     0.84            3.3

4     Twitch.GryphonRB     1136282       1327     6.32           30.0

5      Twitch Kayotica     2919037       1310     3.23            9.7

...                ...         ...        ...      ...            ...

1430          im bloom      106294         26     2.03            3.5

1431     Twitch kaOzs_      803918        396     3.42           13.2

1432           slxyher      603939        310     1.69            5.1

1433         RD-Antony      747295         71     1.10            1.9

1434       O-HO-HO-HO!     1378689        161     1.68            2.3


      Solo matches  Solo kills  Solo minutesPlayed
0            10150    2.297725            1.072340
1             5817    2.262034           -0.541258
3             3687   -0.448978           -0.077769
4             4429    0.918150            0.011352
5            13438    2.528767            1.433994
...            ...         ...                 ...
1430           736   -0.577976           -0.735434
1431          3001    0.037834           -0.197114
1432          6035    0.101137           -0.357635
1433          3763   -0.363007           -0.108463
1434          7105    0.264507            0.512585

[1378 rows x 8 columns]
```

```
plt.scatter(df_sin_outliers['Solo minutesPlayed'],
df_sin_outliers['Solo score'])
plt.xlabel('Solo minutesPlayed')
plt.ylabel('Solo score')
plt.title('Relación entre Solo minutesPlayed y Solo score (Sin
Outliers)')
plt.show()
```



Sin los outliers, podemos observar una tendencia lineal clara y más concentrada entre los minutos jugados en solitario (SoloMinutesPlayed)) y la puntuación (SoloScore) obtenida. A medida que el número de minutos jugados en solitario aumenta, también lo hace la puntuación en solitario.

```
# Seleccionar solo las columnas numéricas
df_numeric = df.select_dtypes(include=[float, int])

# Obtener correlaciones de cada característica en el conjunto de datos
corrmat = df_numeric.corr()
top_corr_features = corrmat.index

plt.figure(figsize=(20, 20))

# Graficar mapa de calor
g = sb.heatmap(df_numeric[top_corr_features].corr(), annot=True,
cmap="RdYlGn")
plt.show()
```



```python
def handling_correlation(df_numeric, threshold=0.8):
  corr_features = set()
  corr_matrix = df_numeric.corr()
  for i in range(len(corr_matrix.columns)):
      for j in range(i):
          if abs(corr_matrix.iloc[i, j]) > threshold:
              colname = corr_matrix.columns[i]
              corr_features.add(colname)
  return list(corr_features)
```

```
correlated_features = handling_correlation(df_numeric, threshold=0.8)

print("Características correlacionadas:", correlated_features)
```

```
Características correlacionadas: ['Solo matches', 'Solo winRatio',
'Solo minutesPlayed']
```

Created in Deepnote