CODER HOUSE

Data Science

Comisión 14075

Profesor: David Bustos Usta

CODER HOUSE

Índice

PRESENTACION

TEMATICA

El mundo de los videojuegos y del streaming de las partidas virtuales encontró en Twitch el entorno perfecto para crecer.
En la actualidad, el auge de esta plataforma se traduce en el aumento constante y creciente de creadores de contenido que abren sus canales, y la gran cantidad de seguidores que acumulan y el número de visitas que logran sumar a diario.
En Twitch, el numero de seguidores es importante, pero lo que realmente influye es el tiempo de transmisión de los stream, y su consecuente impacto en los sponsors para la monetización de las publicidades.
Contar con patrocinadores es uno de los mayores espaldarazos que puede tener tu cuenta de Twitch para ingresar dinero. Serán marcas y servicios que respalden y confíen en tu canal difundiéndolo por otros ámbitos.
Los patrocinios ofrecen un carácter más personalizado y con una mayor capacidad de interacción respecto a la publicidad porque permiten trabajar en base a campañas especializadas.

OBJETIVO

Conocer las características personales de los mejores jugadores/streamers que tiene la plataforma.
Describir y analizar las variables en comparación al genero, peso y las horas transmitidas para determinar la relación entre ellas, y el sedentarismo subyacente que esto conlleva, y adicionalmente quienes serán los mas buscados por los patrocinadores.

EQUIPO

Giuliana Mazzarella: Analista de Datos
Patricia Ruiz Martinez: Analista BI
Guillermo Bensunsan: Analista funcional

DATA WRANGLING

Una vez incorporado el dataframe, identificamos las variables nulas (NaN), detectamos la cantidad de estas, y contando los valores faltantes en cada columna y usando un bucle FOR en Python, pudimos obtener el numero de variables faltantes en cada columna: “True” representando el valor faltante, y “False” significa que el valor esta presente en el conjunto de datos.

En el cuerpo del bucle FOR, el método “.value_counts()” cuenta el numero de valores “True!.

Tomamos la decisión de eliminar las columnas “Unname:15”, “Unname:16”, “Unname:17”, “Unname:18” porque no aportan valor al análisis:

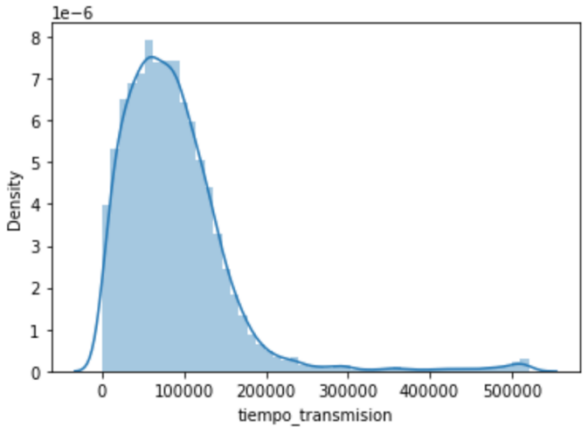
Df.drop([‘Unname:16’, ‘Unname:17’, ‘Unname:18’, ‘Unname:19’], axis=1)

EDA

Con el dataset que disponemos, vamos a analizar cada variable y definir la variable objetivo (de ahora en adelante llamada “VO”) “tiempo_transmision”. El motivo por el cual decidimos tomar esta variable es porque va en concordancia con el objetivo planteado.

UNIVARIADO

Respecto a los graficos univariados, tomamos la VO mencionada y la plasmamos en un histograma:



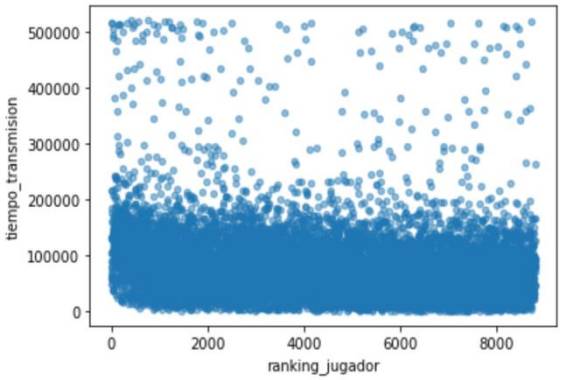
Podemos apreciar una desviación respecto a la distribución normal, una asimetría positiva, y algunos picos aislados.

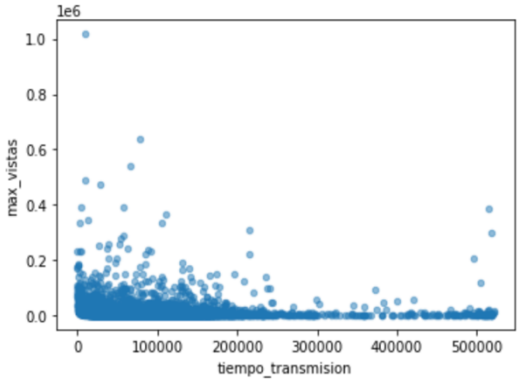
Analizamos también la asimetría y la curtosis de la VO. Podemos ver que la asimetría es de 2.76 y la curtosis de 12.7

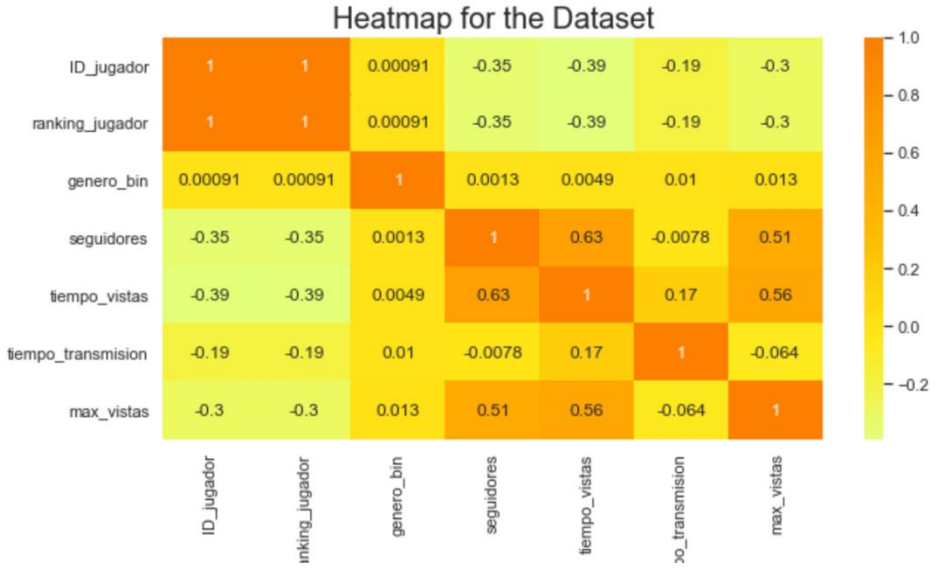
BIVARIADO

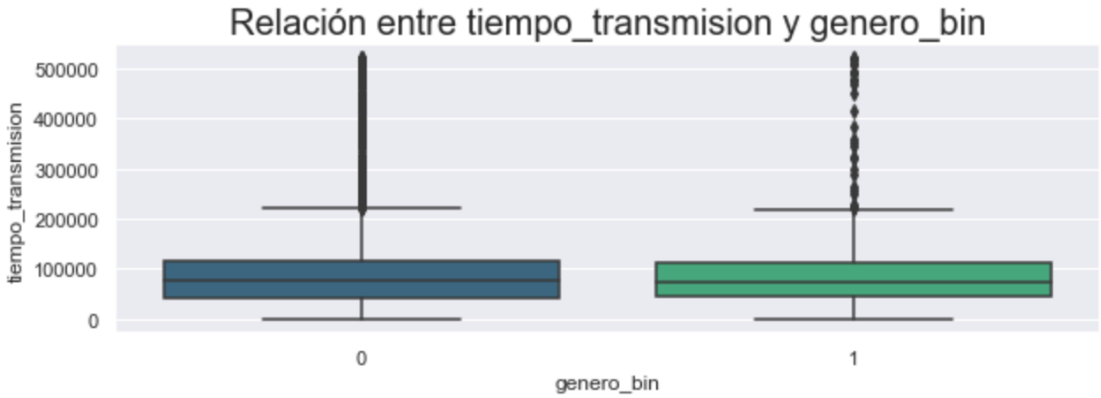
Tomamos la VO con “ranking_jugador” y “max_vistas”, y lo plasmamos en 2 gráficos de dispersión y en 2 box splot.

En ambos gráficos de dispersión se concentra el mayor numero de casos al inicio, y es lineal.









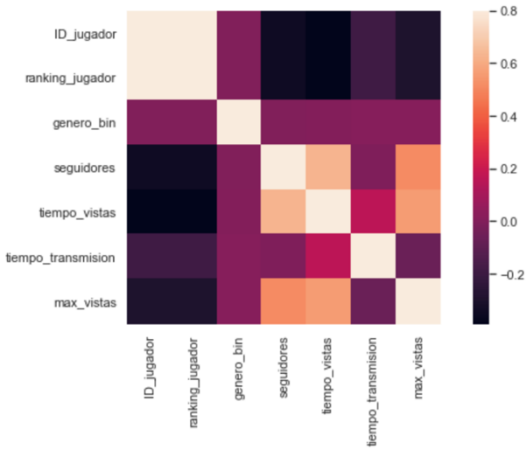
MULTIVARIADO

Hasta ahora sólo me he dejado llevar por la intuición para el análisis de las variables que he creído importantes. Es hora de un análisis más objetivo.

Para ello vamos a realizar las siguientes pruebas de correlación:

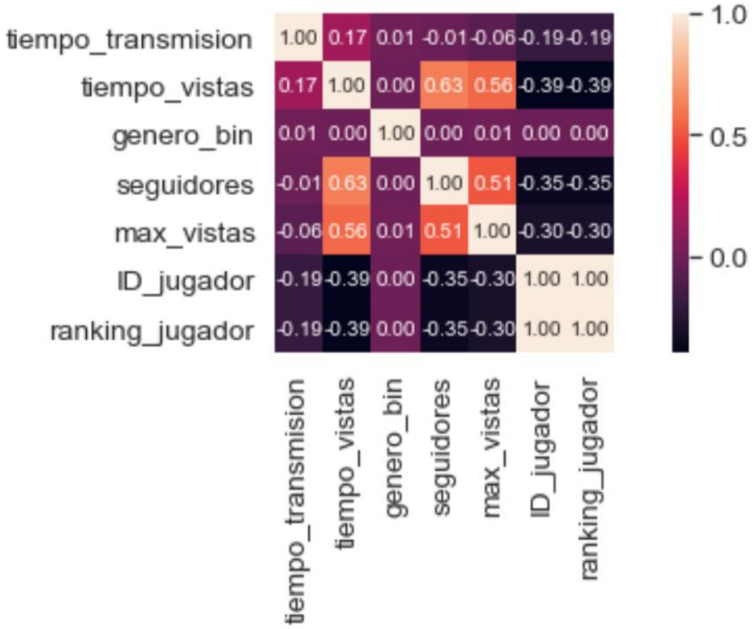
- Matriz de correlación general.
- Matriz de correlación centrada en la variable 'tiempo_transmision'.
- Diagramas de dispersión entre las variables más correladas.

El mapa de calor es una forma visual muy útil para conocer las variables y sus relaciones.

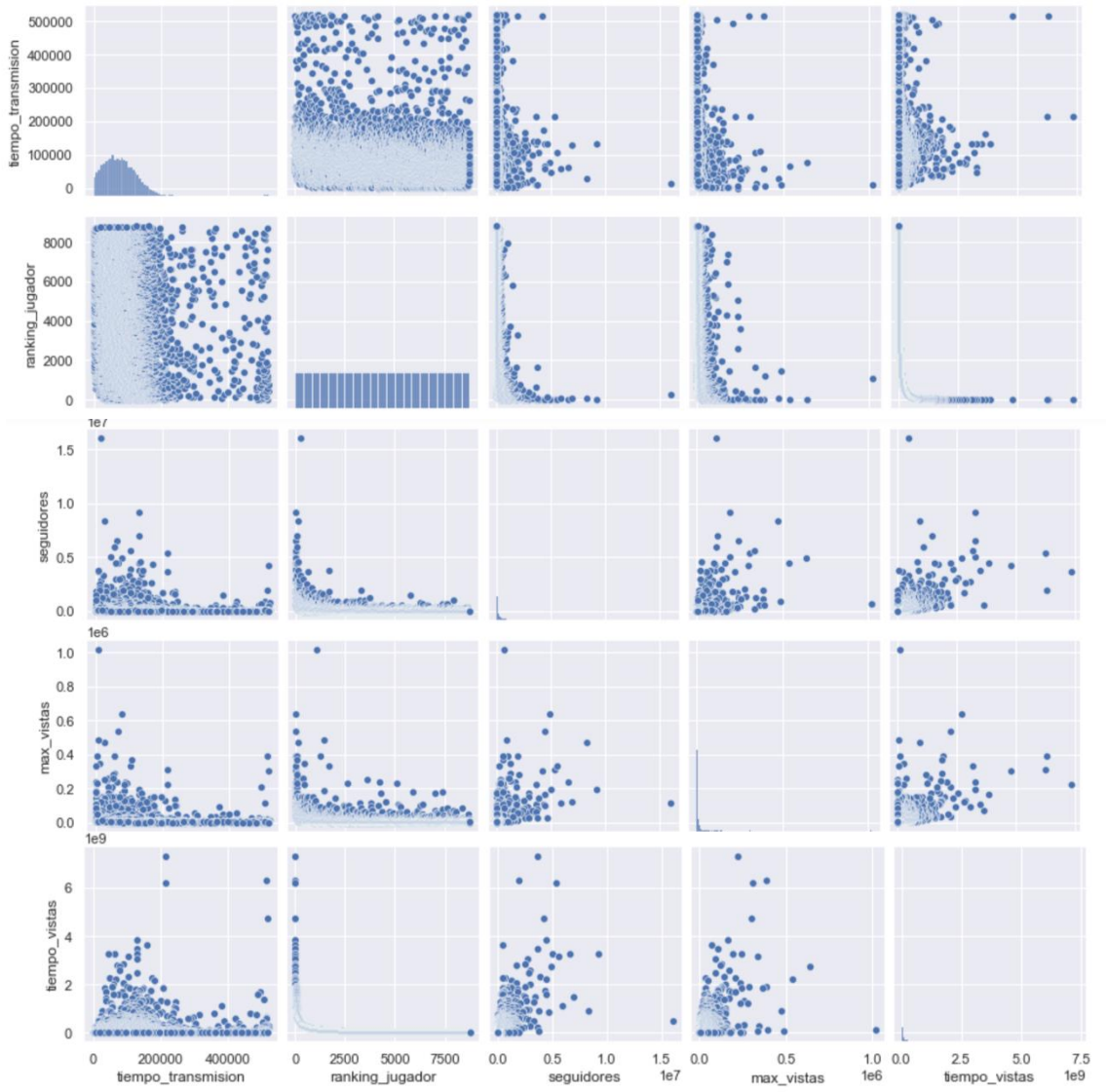


A primera vista, hay dos variables que llaman la atención:

- Las variables relacionadas son id_jugador con ranking: identifican multicolinealidad porque ofrecen la misma información.
- Correlacion entre el “max_vistas”y “seguidores”



Al igual que los graficos anteriores, podemos ver que el tiempo_vistas esta estrechamente relacionado con el “tiempo_transmision”, al igual que “genero_bin”



- El numero de visitas se relaciona con la de “seguidores” y a medida que transcurre el tiempo, aquellas van disminuyendo.
- “max_vistas” también esta ligeramente correlada con “seguidores”

ARBOL DE DECISION

Calculamos el índice de Gini y de entropía, y comparamos cual predecía mejor. Los mismos se podrán verificar a través de los valores dados (ver notebook).

Pero, imaginamos que queremos predecir si la persona es, o no, obesa. Según la descripción del dataset, las personas con indice de 4 o 5 son obesas, por lo que podríamos crear una variable que refleje esto:

data['obese'] = (data.Index >= 4).astype('int').astype('str')

data.drop('Index', axis = 1, inplace = True)

En ese caso, un árbol de decisión nos diría distintas reglas, como por ejemplo, que si el peso de la persona es superior a 100kg, lo más probable es que esa persona sea obesa. Sin embargo, ese corte no será preciso: habrá personas que pesen 100kg que no sean obesas. Así pues, el árbol de decisión sigue creando más ramas que generan nuevas condiciones para ir “afinando” nuestras predicciones y minimizando las impureza (la impureza se refiere a que, cuando hacemos un corte, sea probable que una variable sea clasificada de forma incorrecta)

Por lo que siempre que “cortemos” una variable y la clasificación no sea perfecta, se trata de un corte impuro.

ENTROPIA E INDICE DE GINI

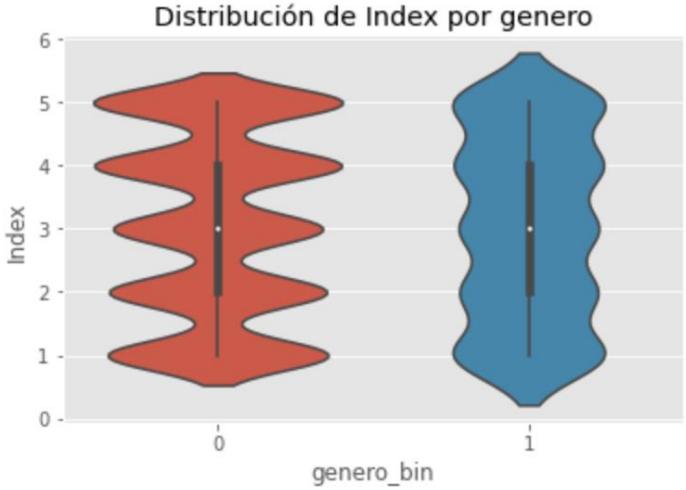
Es una forma de medir la impureza o aleatoriedad en los puntos de datos.

Para medir la entropía se utiliza la probabilidad de una clase.

A diferencia del índice de Gini, cuyo rango va de 0 a 0,5 el rango de la entropía es diferente, ya que va de 0 a 1. De esta forma, los valores cercanos a cero son menos impuros que aquellos que se acercan al 1.

Como vemos, nos da un valor muy cercano al 1, lo cual deducimos que es una impureza similar al índice de impureza de Gini, cuyo valor es cercano al 0,5.

ALGORITMO DE REGRESION




```
# T-test entre clases
# ==============================================================
res_ttest = ttest_ind(
                x1 = datos.Index[datos.genero_bin == 0],
                x2 = datos.Index[datos.genero_bin == 1],
                alternative='two-sided'
            )
print(f"t={res_ttest[0]}, p-value={res_ttest[1]}")
```

```
t=1.0417245414648284, p-value=0.297568031863251
```

Tanto el gráfico como el t-test muestran evidencias de que existe una diferencia entre el índice de las personas con distinto genero. Esta información es útil para considerar los índices como un buen predictor para el modelo.

```
Optimization terminated successfully.
         Current function value: 0.267581
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 7040
Model:                          Logit   Df Residuals:                     7038
Method:                           MLE   Df Model:                            1
Date:                Sat, 02 Oct 2021   Pseudo R-squ.:               0.0008553
Time:                        13:39:29   Log-Likelihood:                -1883.8
converged:                       True   LL-Null:                       -1885.4
Covariance Type:            nonrobust   LLR p-value:                    0.07251
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -2.3334      0.104    -22.465      0.000      -2.537      -2.130
x1            -0.0565      0.031     -1.796      0.073      -0.118       0.005
==============================================================================
```

El coeficiente estimado para la intersección (Intercept o const) es el valor esperado del logaritmo de odds de que una persona con distintos indices. Los odds son muy bajos e2.3334=0.3848818284590452 , lo que se corresponde con una probabilidad de obtener matrícula de p=e0.00012541+e0.0001254=2.9388813302259806 .

Acorde al modelo, el logaritmo de odds de que una persona tenga distinto índice está positivamente relacionado con los índices obtenidos por genero (coeficiente de regresión = -0.0565). Esto significa que, por cada unidad que se incrementa la variable Índice, se espera que el logaritmo de odds de la variable genero_bin se disminuya en promedio 0.0565 unidades.

Aplicando la inversa del logaritmo natural (e-0.0565=0.9450664844951467) se obtiene que, por cada unidad que se incrementa la variable matemáticas, los odds de obtener matrícula se incremente en promedio 0.9450 unidades. No hay que confundir esto último con que la probabilidad de genero_bin se incremente un 0.9450 %.

A diferencia de la regresión lineal en la que $\beta 1$ se corresponde con el cambio promedio en la variable dependiente y debido al incremento en una unidad del predictor $x1$, en regresión logística, $\beta 1$ indica el cambio en el logaritmo de odds debido al incremento en una unidad de $x1$, o lo que es lo mismo, multiplica los odds por $e\beta 1$.

Dado que la relación entre $p(y=1)$ y x no es lineal, los coeficientes de regresión βp no se corresponden con el cambio en la probabilidad de y asociada con el incremento en una unidad de x . Cuánto se incremente la probabilidad de “y” por unidad de “ x” depende del valor de “x” , es decir, de la posición en la curva logística en la que se encuentre.

Además del valor de las estimaciones de los coeficientes parciales de correlación del modelo, es conveniente calcular sus correspondientes intervalos de confianza.

MODELO PREDICTIVO

Comparación de GBDT y LightGBM GBDT (Gradient Boosting Decision Tree) es un modelo duradero en el aprendizaje automático. Su idea principal es utilizar clasificador débil (árbol de decisión) entrenando iterativamente para obtener el modelo óptimo, El modelo tiene buen efecto de entrenamiento y no es fácil de sobre ajustar y otras ventajas. GBDT se usa ampliamente en la industria y a menudo se usa para predicción de CTR, ranking de búsqueda y otras tareas. GBDT también es un arma letal para varias competiciones de minería de datos.

LightGBM (Light Gradient Boosting Machine) es un marco que implementa el algoritmo GBDT, admite un entrenamiento paralelo eficiente y tiene las siguientes ventajas: Mayor velocidad de entrenamiento Menor consumo de memoria Mejor precisión Soporte distribuido para el procesamiento rápido de datos masivos.

En líneas generales, podemos notar que los stremers de esta plataforma, son obesos, que en su mayoría son hombres, y esta directamente proporcionado en relación a la horas trasmitidas por ellos.

CONCLUSIONES FINALES

El mundo del stream esta en auge, y cada vez mas son los jóvenes y adultos que se suman a esta tendencia.
El sedentarismo, a raíz de la pandemia se multiplico, y con este análisis lo que quisimos comprobar es la relación que hay entre los jugadores que están frente a la computadora y su peso, y su consecuente impacto en el sedentarismo.
A raíz de la elección de los modelos seleccionados, hemos optado por el Gradient Boosting debido a que es el modelo mas certero por el accuracy arrojado (0,9290) y que la predicción de una nueva observación se obtiene agregando las predicciones de todos los árboles individuales que forman el modelo.

Comparando el accuracy de ambos modelos (LightGBM y Gradient Boosting), podemos notar que el que mide el mejor porcentaje de los casos, que el modelo haya acertado, es el de Gradient Boosting con 0,9290. Igualmente, LightGB tiene muy buen accuracy con 0,9265.

Cualquiera de los dos es muy bueno, pero el que mas acierta es Gradiant Boosting.