TRABAJO FINAL

POR:

CAMILO TOBÓN FLOREZ

ESPECIALIZACIÓN EN BIG DATA E INTELIGENCIA DE NEGOCIOS

DOCENTE:
JHON VELEZ

UNIVERSIDAD CATOLICA LUIS AMIGO
MEDELLÍN
2021

Contenido

VENTA DE VIDEO JUEGOS A NIVEL MUNDIAL

Fase 1. definición de necesidades del cliente (Comprensión del negocio)

El mercado de los video juegos ha venido ganando protagonismo en los últimos años, hay una gran variedad de marcas creadores y publicadoras de video juegos, sin embargo, el mercado de las consolas o plataformas es muy oligopólico, ya que unas pocas marcas representan casi la totalidad de la torta en cuanto consolas.

Se desea observar el porcentaje de ventas de video juegos por marca.

Se quiere conocer la correlación de los datos, con el fin de determinar si hay colinealidad entre variables y si hay variables predictoras interesantes para predecir las ventas globales de video juegos.

Se desea observar el comportamiento de los video juegos lanzados en los diferentes años por plataforma y por marca.

Fase 2. Estudio y comprensión de los datos

El dataset `Video_Games_Sales_as_at_22_Dec_2016.csv` está compuesto por 18 columnas y 16.719 registros o filas, consiste en video juegos lanzados desde 1980 hasta el año 2020, con su fecha de lanzamiento, ventas en cantidad en diferentes regiones y a nivel mundial, entre otros. A continuación, se detallan las variables y su descripción.

Variable	Descripción
Name	Nombre del video juego
Platform	Consola en la que funciona el video juego
Year_of_Release	Año de lanzamiento
Genre	Género o categoría del video juego
Publisher	Marca que publica el video juego
NA_Sales	Cantidad de ventas en EEUU
EU_Sales	Cantidad de ventas en Europa
JP_Sales	Cantidad de ventas en Japón
Other_Sales	Cantidad de ventas en otros países
Global_Sales	Cantidad de ventas a nivel global
Critic_Score	Puntaje o calificación otorgada al video juego por parte de la critica
Critic_Count	Conteo de los críticos que calificaron el video juego
User_Score	Puntaje o calificación otorgada al video juego por parte de los usuarios
User_Count	Conteo de los usuarios que calificaron el video juego
Developer	Marca creadora del video juego
Rating	Clasificación del video juego

También se elabora un dataset adicional compuesto por la `Platform` o consola y la marca de esta consola, esto con el fin de agrupar por marca de consola y analizar por esta variable.

Fase 3. Análisis de los datos y selección de características

Se carga el archivo de la facturacion por video juegos y se lleva a la variable ‘df’

```
In [3]: df = pd.read_csv('Video_Games_Sales_as_at_22_Dec_2016.csv')
```

```
In [65]: df.head(2)
```

Out[65]:

	Name	Platform	Year_of_Release	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
0	Wii Sports	Wii	2006.0	Sports	Nintendo	41.36	28.96	3.77	8.45	82.53
1	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24

Se carga un archivo armado manualmente el cual tiene la plataforma o consola y la marca a la que pertenece. Para este caso se utiliza el parámetro ‘sep’ para indicar el delimitador.

```
In [31]: marca_consola = pd.read_csv('consola_marca.csv', sep=';')
```

```
In [66]: marca_consola.head(5)
```

Out[66]:

	Consola	Marca
0	PC	PC
1	GB	Nintendo
2	PS2	PlayStation
3	DS	Nintendo
4	X360	Xbox

Se utiliza la función ‘merge’ de la librería pandas, para cruzar dos tablas o dataframes, se indica las llaves de cruce y el tipo de cruce, en este caso ‘left’, el cual consiste en que la tabla de la izquierda se conserva y se traen los datos que cruzan de la tabla de la derecha.

```
In [33]: df = pd.merge(df, marca_consola, how='left', left_on='Platform', right_on='Consola')

In [67]: df.head(2)

Out[67]:
```

IP_Sales	Other_Sales	Global_Sales	Critic_Score	Critic_Count	User_Score	User_Count	Developer	Rating	Consola	Marca
3.77	8.45	82.53	76.0	51.0	8	322.0	Nintendo	E	Wii	Nintendo
6.81	0.77	40.24	NaN	NaN	NaN	NaN	NaN	NaN	NES	Nintendo

Se ejecuta el método que muestra el tipo de dato que tiene cada columna y cuantos registros 'no nulos' tiene. Esto con el fin de determinar que columnas se van a tener en cuenta para los análisis.

En rojo se marcan las columnas que poseen una gran cantidad de valores nulos.

```
In [68]: df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 16719 entries, 0 to 16718
Data columns (total 18 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Name             16717 non-null  object
 1   Platform         16719 non-null  object
 2   Year_of_Release  16450 non-null  float64
 3   Genre            16717 non-null  object
 4   Publisher        16665 non-null  object
 5   NA_Sales         16719 non-null  float64
 6   EU_Sales         16719 non-null  float64
 7   JP_Sales         16719 non-null  float64
 8   Other_Sales      16719 non-null  float64
 9   Global_Sales     16719 non-null  float64
 10  Critic_Score     8137 non-null   float64
 11  Critic_Count     8137 non-null   float64
 12  User_Score       10015 non-null  object
 13  User_Count       7590 non-null   float64
 14  Developer        10096 non-null  object
 15  Rating           9950 non-null   object
 16  Consola          16719 non-null  object
 17  Marca            16719 non-null  object
dtypes: float64(9), object(9)
memory usage: 2.4+ MB
```

Se genera una descripción estadística de los datos

```
In [69]: df.describe().T
```

Out[69]:

	count	mean	std	min	25%	50%	75%	max
Year_of_Release	16450.0	2006.487356	5.878995	1980.00	2003.00	2007.00	2010.00	2020.00
NA_Sales	16719.0	0.263330	0.813514	0.00	0.00	0.08	0.24	41.36
EU_Sales	16719.0	0.145025	0.503283	0.00	0.00	0.02	0.11	28.96
JP_Sales	16719.0	0.077602	0.308818	0.00	0.00	0.00	0.04	10.22
Other_Sales	16719.0	0.047332	0.186710	0.00	0.00	0.01	0.03	10.57
Global_Sales	16719.0	0.533543	1.547935	0.01	0.06	0.17	0.47	82.53
Critic_Score	8137.0	68.967679	13.938165	13.00	60.00	71.00	79.00	98.00
Critic_Count	8137.0	26.360821	18.980495	3.00	12.00	21.00	36.00	113.00
User_Count	7590.0	162.229908	561.282326	4.00	10.00	24.00	81.00	10665.00

Se generan los cuantiles 0.95, 0.99, 0.992, 0.995, 0,999 y 1 con el fin de determinar si existen outliers que impidan la exploración estadística.

```
In [74]: df[['NA_Sales', 'EU_Sales', 'JP_Sales', 'Other_Sales', 'Global_Sales']].quantile([0.95, 0.99, 0.992, 0.995, 0.999, 1]).T
```

Out[74]:

	0.950	0.990	0.992	0.995	0.999	1.000
NA_Sales	1.06	2.7900	3.14256	4.0500	9.46102	41.36
EU_Sales	0.62	1.9382	2.14256	2.7264	6.13692	28.96
JP_Sales	0.36	1.2682	1.42512	1.9882	4.13846	10.22
Other_Sales	0.20	0.6300	0.72256	0.9341	2.04282	10.57
Global_Sales	2.04	5.4582	6.35280	8.0582	20.87486	82.53

Grafico de Torta por marca de consola

Se genera una agrupación por marca de plataforma y sumatoria de las ventas globales de video juegos, para esta agrupación se genera un gráfico de torta.

Se evidencia que la mayoría de las ventas de video juegos está concentrada en tres marcas de plataformas principales, PlayStation con 40.2%, Nintendo con un 39.3 % y Xbox con un 15.6%

In [153]:
```
ventas_marca = df.groupby('Marca').agg({'Global_Sales':'sum'}).reset_index().sort_values(by='Global_Sales')
plt.figure(figsize=(10, 10))
plt.pie(ventas_marca['Global_Sales'], labels=ventas_marca['Marca'],
        autopct='%1.1f%%', shadow=True, startangle=90, rotatelabels=True)
plt.axis()
plt.show()
```

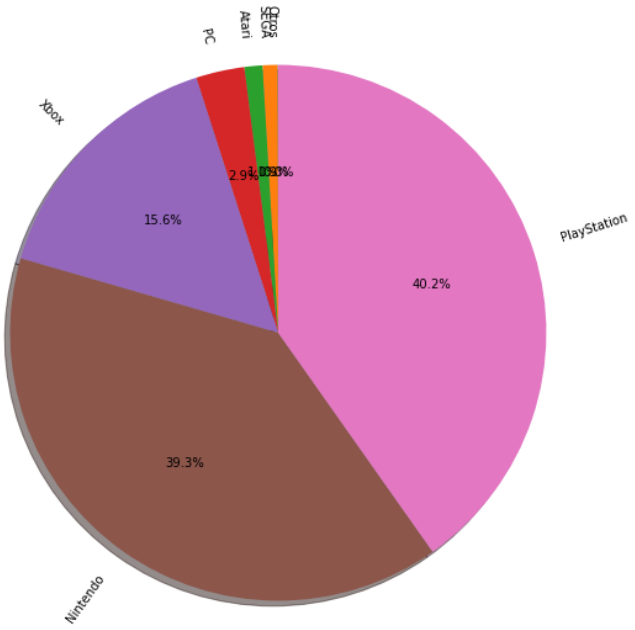


Grafico de histogramas por región de ventas

Utilizando un ciclo `for` se generan histogramas para las diferentes variables de ventas de video juegos, vemos que es asimétrico hacia la derecha.

In [99]:
```
plt.figure()
nbins = 100
for i in ventas:
    sns.histplot(df[i[0]], bins=nbins)
    plt.title('Histograma ventas ' + i[1])
    plt.show()
```

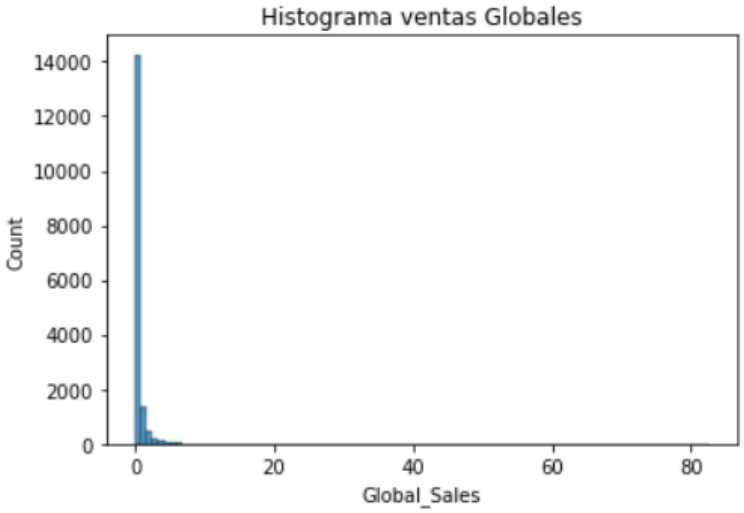


Gráfico de violín por marca de consola

Se genera un gráfico de violín de ventas de video juegos por marca de consola, en este caso no se toman todos los valores, sino el 99.5% de ellos, con el fin de excluir los valores mayores y así poder visualizar correctamente.

Se evidencia que las ventas de video juegos por marca de consola, muestra una gran densidad (frecuencia) en los valores cercanos bajos.

```
df_resumido = df[df.Global_Sales < 22]
plt.figure(figsize=(12, 6))
sns.violinplot(x='Marca', y='Global_Sales', data=df_resumido)
```

```
<AxesSubplot:xlabel='Marca', ylabel='Global_Sales'>
```

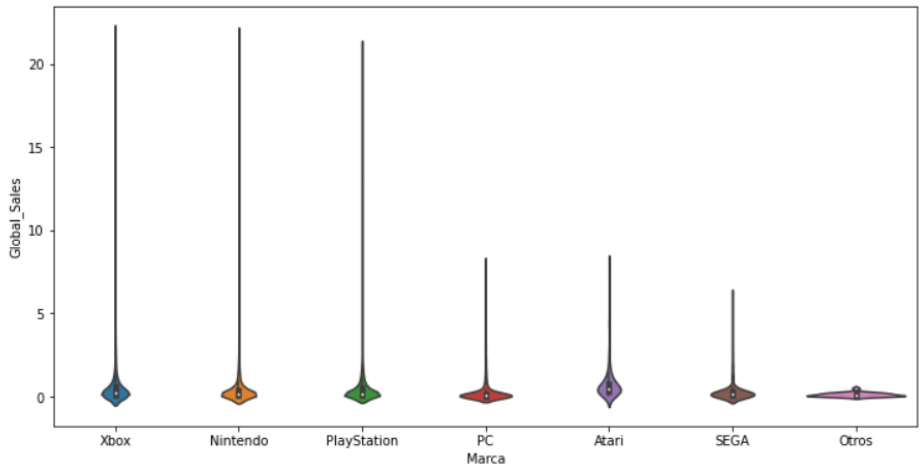


Gráfico de dispersión

Se evidencia que en la marca de consolas `Nintendo` hay video juegos con grandes ventas, `PlayStation` y `Xbox` tienen un comportamiento muy similar, en estas dos marcas hay algunos video juegos que destacan en ventas sobre los demás.

```
plt.figure()
plt.scatter(x='Global_Sales', y='Marca', data=df, label="s")
```

```
<matplotlib.collections.PathCollection at 0x1ce52fa5040>
```

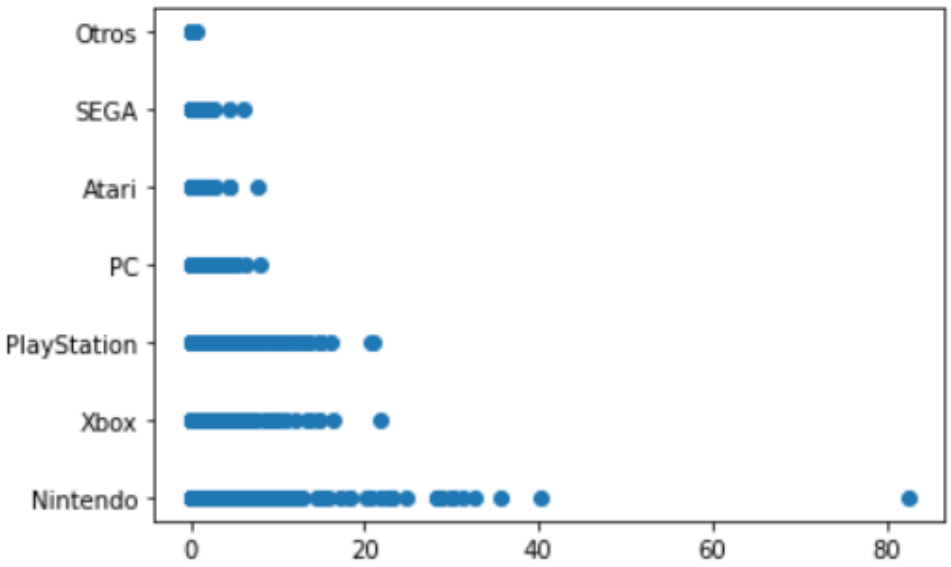


Gráfico de tendencia

Se genera un gráfico de tendencia de ventas globales de video juegos por marca de consola.

Se logra ver que para la marca `Nintendo`, hay video juegos con gran cantidad de ventas, pero sin tanta densidad en esas zonas, para `Xbox` y `PlayStation`, no hay tantos video juegos con ventas tan elevadas.

```
sns.stripplot(x="Global_Sales", y="Marca", data=df)
```

```
<AxesSubplot:xlabel='Global_Sales', ylabel='Marca'>
```

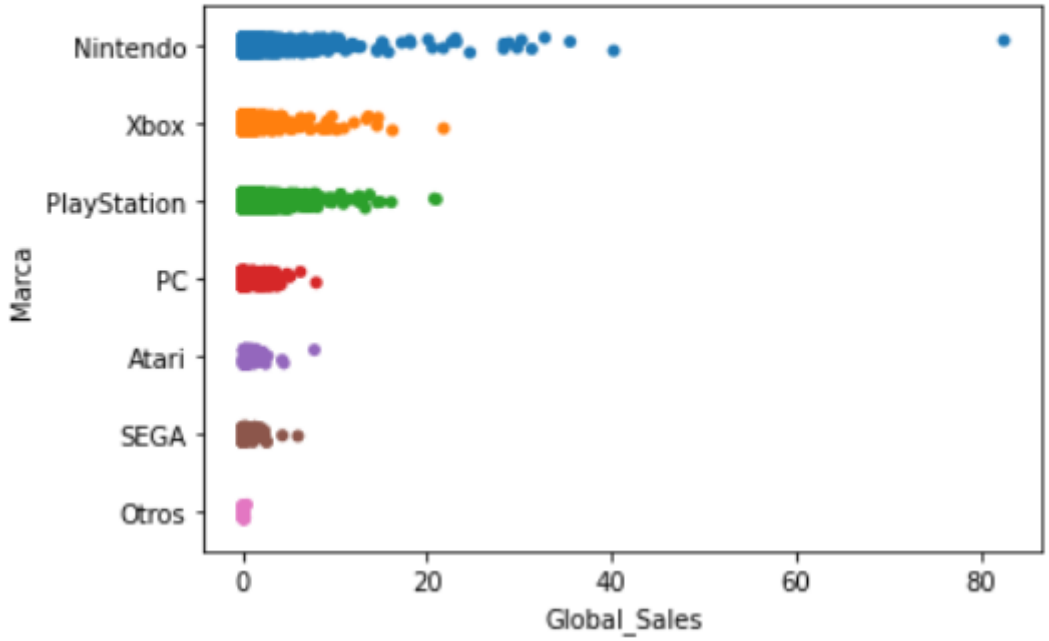
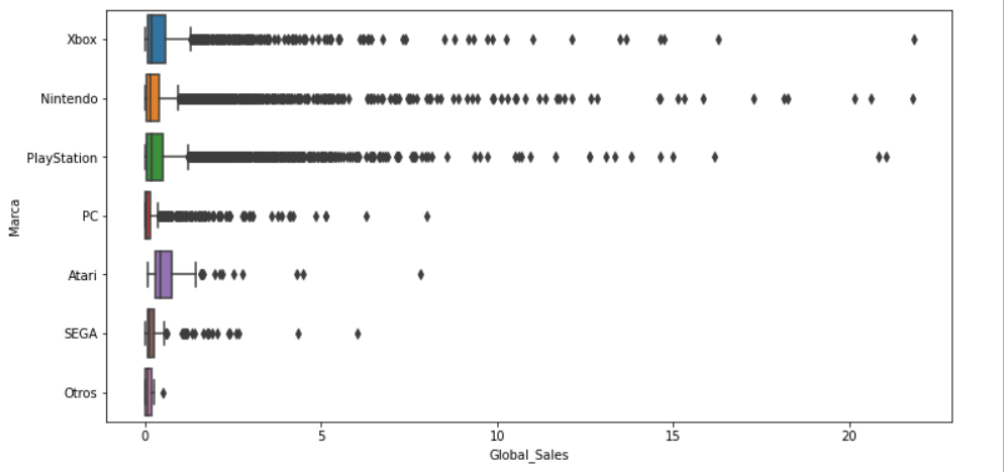


Gráfico de bigotes o boxplot

En las 3 marcas principales de consolas, el promedio de ventas de video juegos es muy similar, según la gráfica, `Nintendo` y `PlayStation` tienen mayor número de video juegos con altas ventas.

```
plt.figure(figsize=(12,6))
sns.boxplot(x='Global_Sales', y='Marca', data=df_resumido)
plt.title('')
plt.show()
```

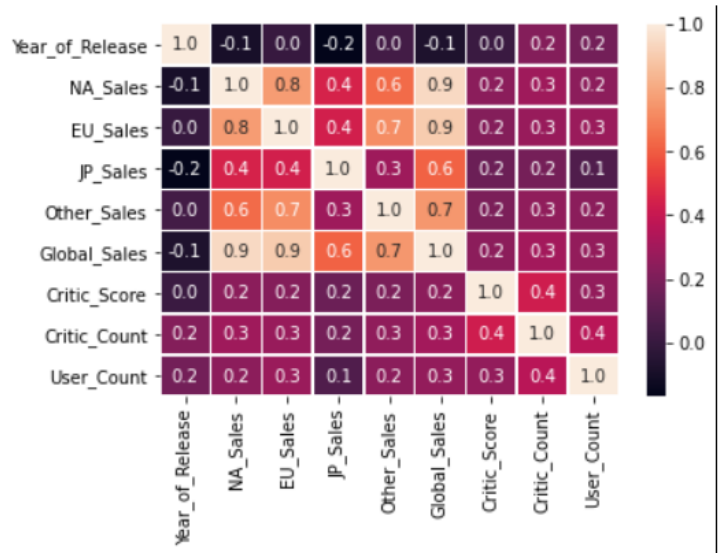


Correlación

Se genera un mapa de calor con la correlación entre las variables numéricas, las diferentes variables de ventas están altamente correlacionadas entre sí, lo que muestra una colinealidad entre las variables, por tal razón no se podrían utilizar como predictores.

```
sns.heatmap(df.corr(), annot=True, linewidth=0.5, fmt='.1f')
```

```
<AxesSubplot:>
```



Se genera una agrupación por ‘año de lanzamiento’ y ‘plataforma’ y se totaliza por ‘ventas globales’, ‘ventas en EEUU’, ‘ventas en Europa’, ‘ventas en Japón’ y ‘otras ventas’.

```
In [75]: anho_plataforma = df.groupby(['Year_of_Release', 'Platform']).agg(
             {'Global_Sales':'sum', 'NA_Sales':'sum', 'EU_Sales':'sum', 'JP_Sales':'sum', 'Other_Sales':'sum'}).reset_index()
```

```
In [77]: anho_plataforma.head()
```

Out[77]:

	Year_of_Release	Platform	Global_Sales	NA_Sales	EU_Sales	JP_Sales	Other_Sales
0	1980.0	2600	11.38	10.59	0.67	0.0	0.12
1	1981.0	2600	35.77	33.40	1.96	0.0	0.32
2	1982.0	2600	28.86	26.92	1.65	0.0	0.31
3	1983.0	2600	5.83	5.44	0.34	0.0	0.06
4	1983.0	NES	10.96	2.32	0.46	8.1	0.08

También se genera una agrupación por ‘año de lanzamiento’ y ‘marca’ y se totaliza por las mismas variables del ejemplo anterior.

```
In [ ]: anho_marca = df.groupby(['Year_of_Release', 'Marca']).agg(
            {'Global_Sales':'sum', 'NA_Sales':'sum', 'EU_Sales':'sum', 'JP_Sales':'sum', 'Other_Sales':'sum'}).reset_index()
```

```
In [80]: anho_marca.sample(5)
```

Out[80]:

	Year_of_Release	Marca	Global_Sales	NA_Sales	EU_Sales	JP_Sales	Other_Sales
19	1990.0	Nintendo	46.79	23.60	7.08	14.77	1.32
99	2008.0	Xbox	135.43	82.64	38.20	1.89	12.71
5	1984.0	Atari	0.27	0.26	0.01	0.00	0.00
126	2015.0	PlayStation	142.18	47.75	58.49	14.95	20.90
32	1994.0	PlayStation	6.02	1.76	1.20	2.67	0.40

Se crean dos listas para la graficación, en la primer lista se definen las plataformas, el color y el tipo de linea, en la segunda lista se define el nombre de columna en el dataframe y el titulo que se va a graficar

```
In [81]: plataforma_color_simbolo = [['X360', 'b', '--'],
                                     ['XOne', 'g', '--'],
                                     ['XB', 'y', '--'],
                                     ['PS', 'm', '-'],
                                     ['PS2', 'r', '-'],
                                     ['PS3', 'c', '-'],
                                     ['PS4', 'k', '-']]
```

```
In [82]: ventas = [['Global_Sales', 'Globales'],
                  ['NA_Sales', 'EEUU'],
                  ['EU_Sales', 'Europa'],
                  ['JP_Sales', 'Japon'],
                  ['Other_Sales', 'Otros paises']]
```

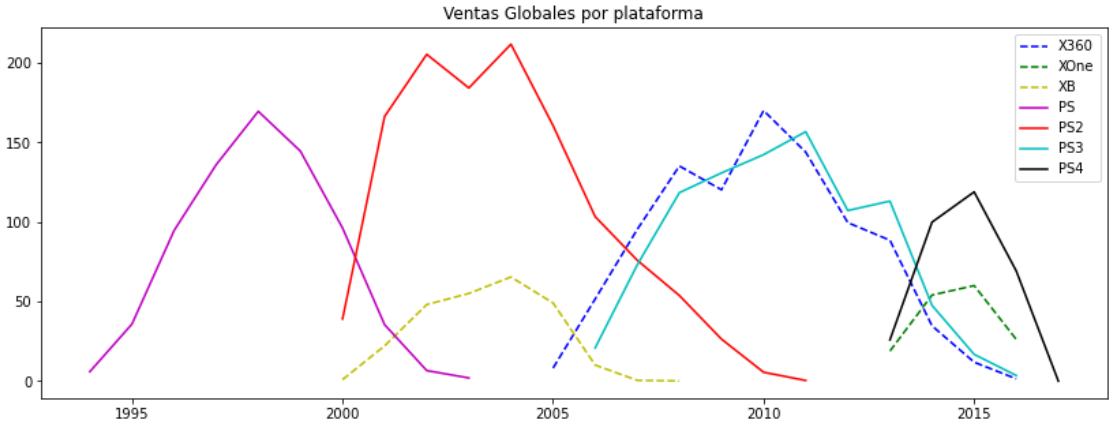
Gráfico de líneas – histórico de ventas de video juegos por plataforma o consola

Se genera un ciclo `for`, con el fin de recorrer la lista que tiene los títulos de las columnas de ventas, después se declara otro ciclo `for` para graficar una línea por cada una de las plataformas, por último, se genera el título y las leyendas para cada uno de los gráficos.

```
In [94]: for x, y in enumerate(ventas):
             region = y[0]
             region_titulo = y[1]
             plt.figure(figsize=(16, 6))
             for i, j in enumerate(plataforma_color_simbolo):
                 plt.plot(anho_plataforma.Year_of_Release[anho_plataforma['Platform']==j[0]],
                          anho_plataforma[region][anho_plataforma['Platform']==j[0]],
                          color=j[1], linestyle=j[2], label=j[0]
                         )
             titulo = 'Ventas '+ region_titulo + ' por plataforma'
             plt.legend(['X360', 'XOne', 'XB', 'PS', 'PS2', 'PS3', 'PS4'])
             plt.title(titulo)
             plt.show
```

En el grafico podemos observar que cada que se lanza una nueva consola de la misma marca, el lanzamiento de video juegos de la consola anterior tiene un declive acelerado.

Las fechas de lanzamiento de los video juegos para ambas marcas PlayStation y Xbox por lo general son por las mismas fechas, lo cual denota una fuerte competencia.



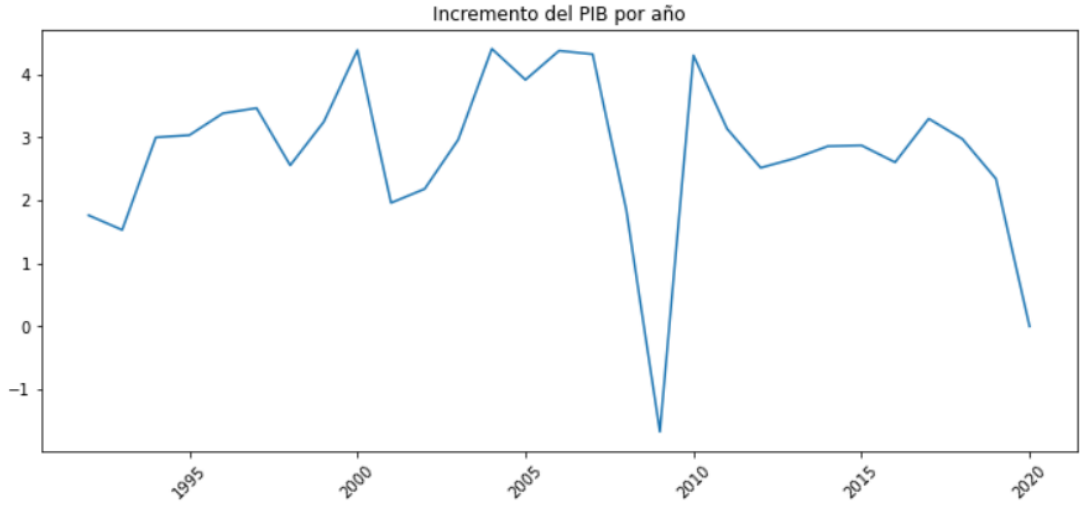
Relación de la economía con la venta de video juegos en el mundo

Se carga un archivo con el crecimiento del PIB (Producto interno bruto) por países, regiones y global.

```
economia = pd.read_csv('economia/API_NY.GDP.MKTP.KD.ZG_DS2_es_csv_v2_2253924.csv', header=2)
economia.drop(['Country Code', 'Indicator Code', 'Indicator Name', '1960', '1961', '1962',
               '1963', '1964', '1965', '1966', '1967', '1968','1969', '1970', '1971',
               '1972', '1973', '1974', '1975', '1976', '1977',
               '1978', '1979', '1980', '1981', '1982', '1983', '1984', '1985', '1986',
               '1987', '1988', '1989', '1990', '1991','Unnamed: 65'], axis=1, inplace=True)
economia = economia.fillna(0)
economia_global = economia[economia['Country Name']=='Mundo'].T.reset_index()
economia_global = economia_global.drop([0], axis=0)
economia_global.columns = ['año', 'incremento_PIB']
economia_global.año = economia_global.año.astype(int)
```

Se crea un gráfico de líneas, para mostrar el comportamiento del PIB a nivel mundial

```
plt.figure(figsize=(12, 5))
plt.plot(economia_global.año,
         economia_global.incremento_PIB[economia_global.año >= 1992])
plt.xticks(rotation=45)
plt.title('Incremento del PIB por año')
plt.show()
```



Se realiza una agrupación por marca y se sumariza por las diferentes variables de ventas

```
anho_marca = df.groupby(['Year_of_Release', 'Marca']).agg({'Global_Sales':'sum',
    'NA_Sales':'sum', 'EU_Sales':'sum', 'JP_Sales':'sum', 'Other_Sales':'sum'}).reset_index()
```

```
anho_marca
```

	Year_of_Release	Marca	Global_Sales	NA_Sales	EU_Sales	JP_Sales	Other_Sales
0	1980.0	Atari	11.38	10.59	0.67	0.00	0.12
1	1981.0	Atari	35.77	33.40	1.96	0.00	0.32
2	1982.0	Atari	28.86	26.92	1.65	0.00	0.31
3	1983.0	Atari	5.83	5.44	0.34	0.00	0.06
4	1983.0	Nintendo	10.96	2.32	0.46	8.10	0.08

Se realiza el escalamiento de las variables ventas globales de video juegos y crecimiento año a año del PIB, con el fin de poder ver la comparativa visualmente.

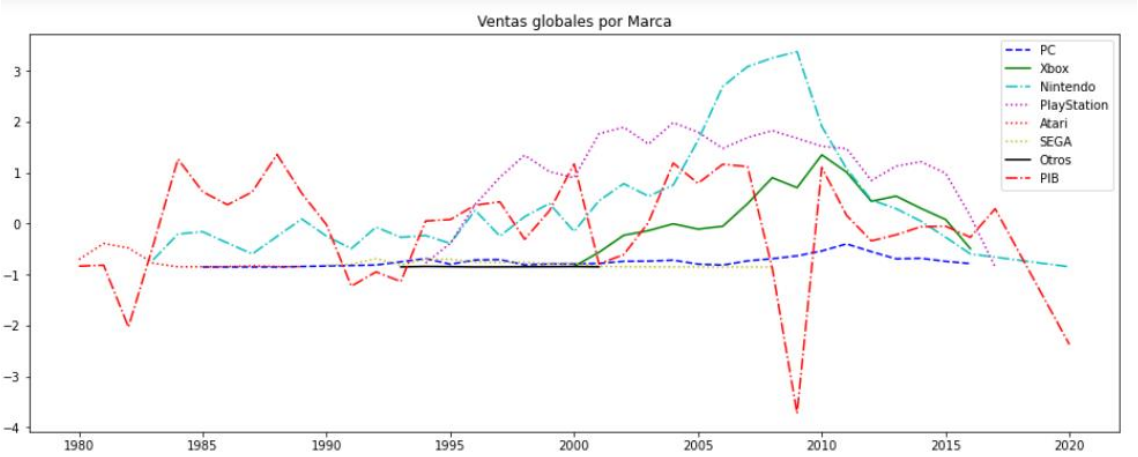
```
anho_marca.fillna(0, inplace=True)
scaler = StandardScaler()
```

```
anho_marca['Global_Sales_scaled'] = scaler.fit_transform(anho_marca[['Global_Sales']]).reshape(-1, 1)
anho_marca['incremento_PIB_scaled'] = scaler.fit_transform(anho_marca[['incremento_PIB']]).reshape(-1,
```

Se genera una gráfica de línea para observar el comportamiento de las ventas globales de video juegos por marca de consola e incremento del PIB por año. Esto con el fin de buscar patrones entre ambas variables.

```
plt.figure(figsize=(16, 6))
for i, j in enumerate(marca_color_simbolo):
    plt.plot(anho_marca.Year_of_Release[anho_marca['Marca']==j[0]],
             anho_marca.Global_Sales_scaled[anho_marca['Marca']==j[0]],
             color=j[1], linestyle=j[2], label=j[0]
            )

plt.plot(anho_marca.Year_of_Release,
             anho_marca.incremento_PIB_scaled,
             color='r', linestyle='-.', label='PIB'
            )
plt.legend(['PC', 'Xbox', 'Nintendo', 'PlayStation', 'Atari', 'SEGA', 'Otros', 'PIB'])
plt.title('Ventas globales por Marca')
plt.show
```



Fase 4. Modelado

Se borran los registros que no tienen los datos de críticas, con el fin de utilizar esas variables para utilizar un modelo predictivo de regresión. Se selecciona un modelo de regresión, ya que la variable a predecir es numérica continua.

```
In [41]: df_calificacion = df.dropna()
```

Se separa el dataset en dos conjuntos de datos:

* dataset `X` que corresponde a las variables predictoras, se excluyen las variables asociadas a ventas, debido a que existe una correlación superior al 80% con la variable a predecir de `Global_Sales`, existe colinealidad.
* dataset `y` que corresponde a la variable a predecir `Global_Sales`

También se utiliza el método `get_dummies` el cual sirve para convertir las variables categóricas a numéricas, este metodo realiza un `One-Hot Encode`.

```
X = pd.get_dummies(df_calificacion[['Platform', 'Year_of_Release', 'Genre', 'Publisher',
                    'Critic_Score', 'Critic_Count', 'User_Count',
                   'Developer', 'Rating', 'Marca']])
y = df_calificacion['Global_Sales']
```

Se utiliza el método `train_test_split` el cual divide aleatoriamente los conjuntos de datos, con el fin de generar un conjunto de datos de entrenamiento y otro de pruebas.

En este caso se indicó que el conjunto de prueba es el 20% del total de los datos.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Regresión Lineal

Se define el modelo de regresión lineal, posteriormente se entrena y por ultimo se predice la variable de salida para el conjunto de datos de prueba.

```
regr = lm.LinearRegression(normalize=True)
regr.fit(X_train, y_train)
y_pred = regr.predict(X_test)
```

Random Forest Regressor - Bosques Aleatorios para Regresión

Se crea un Bosque de árboles aleatorios, el cual es un método de ensamble bastante robusto, que combina varios árboles de decisión.

el algoritmo entrena cada modelo secuencialmente con todos los datos y, para cada nuevo modelo, se le da más peso a los datos que no fueron bien clasificados o cuyo error en regresión sea más alto.

Se complemento con el método `GridSearchCV` la sirve para buscar los mejores parámetros para el modelo, para cada parámetro se indica una lista con posibles valores, este método seleccionara los mejores.

```
%%time
forest_est = RandomForestRegressor()

param_grid = {'n_estimators': [5, 10, 20, 30, 40, 50, 80, 100],
              'min_samples_leaf': [10, 50, 100, 1000],
              'max_depth': [5, 10, 15, 20, 30]}

forest_grid = GridSearchCV(forest_est, param_grid=param_grid, cv=4)

forest_grid.fit(X_train, y_train)
```

```
Wall time: 41min 33s

GridSearchCV(cv=4, estimator=RandomForestRegressor(),
             param_grid={'max_depth': [5, 10, 15, 20, 30],
                         'min_samples_leaf': [10, 50, 100, 1000],
                         'n_estimators': [5, 10, 20, 30, 40, 50, 80, 100]})
```

```
forest_grid.best_params_
```

```
{'max_depth': 30, 'min_samples_leaf': 10, 'n_estimators': 30}
```

```
y_predict_grid = forest_grid.predict(X_test)
```

Fase 5. Evaluación

Se evalúa el modelo de regresión lineal, pero obtiene resultados muy bajos.

```
print('Mean Squared Error:', mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test, y_pred)))
print('R-squared score (training): {:.3f}'
 .format(regr.score(X_train, y_train)))
print('R-squared score (test): {:.3f}\n'
 .format(regr.score(X_test, y_test)))
```

```
Mean Squared Error: 1.299867681836884e+29
Root Mean Squared Error: 360536777851703.2
R-squared score (training): 0.279
R-squared score (test): -47830369015731240811093819392.000
```

Se utilizan las mismas métricas utilizadas en la regresión lineal. se observa mejores resultados.
Para el conjunto de datos de entrenamiento tenemos un resultado de 0.521 y para los datos de prueba tenemos 0.466

```
print('Mean Absolute Error:', mean_absolute_error(y_test
                                                  , y_predict_grid))
print('Mean Squared Error:', mean_squared_error(y_test, y_predict_grid))
print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test, y_predict_grid)))
print('R-squared score (training): {:.3f}'
 .format(forest_grid.score(X_train, y_train)))
print('R-squared score (test): {:.3f}\n'
 .format(forest_grid.score(X_test, y_test)))
```

```
Mean Absolute Error: 0.48083573099180277
Mean Squared Error: 1.4524784234896848
Root Mean Squared Error: 1.205188127841328
R-squared score (training): 0.521
R-squared score (test): 0.466
```

Se selecciona el método Random Forest Regressor, debido a que se obtienen mejores resultados, sin embargo, cabe destacar que es un modelo bastante robusto que es costoso computacionalmente.

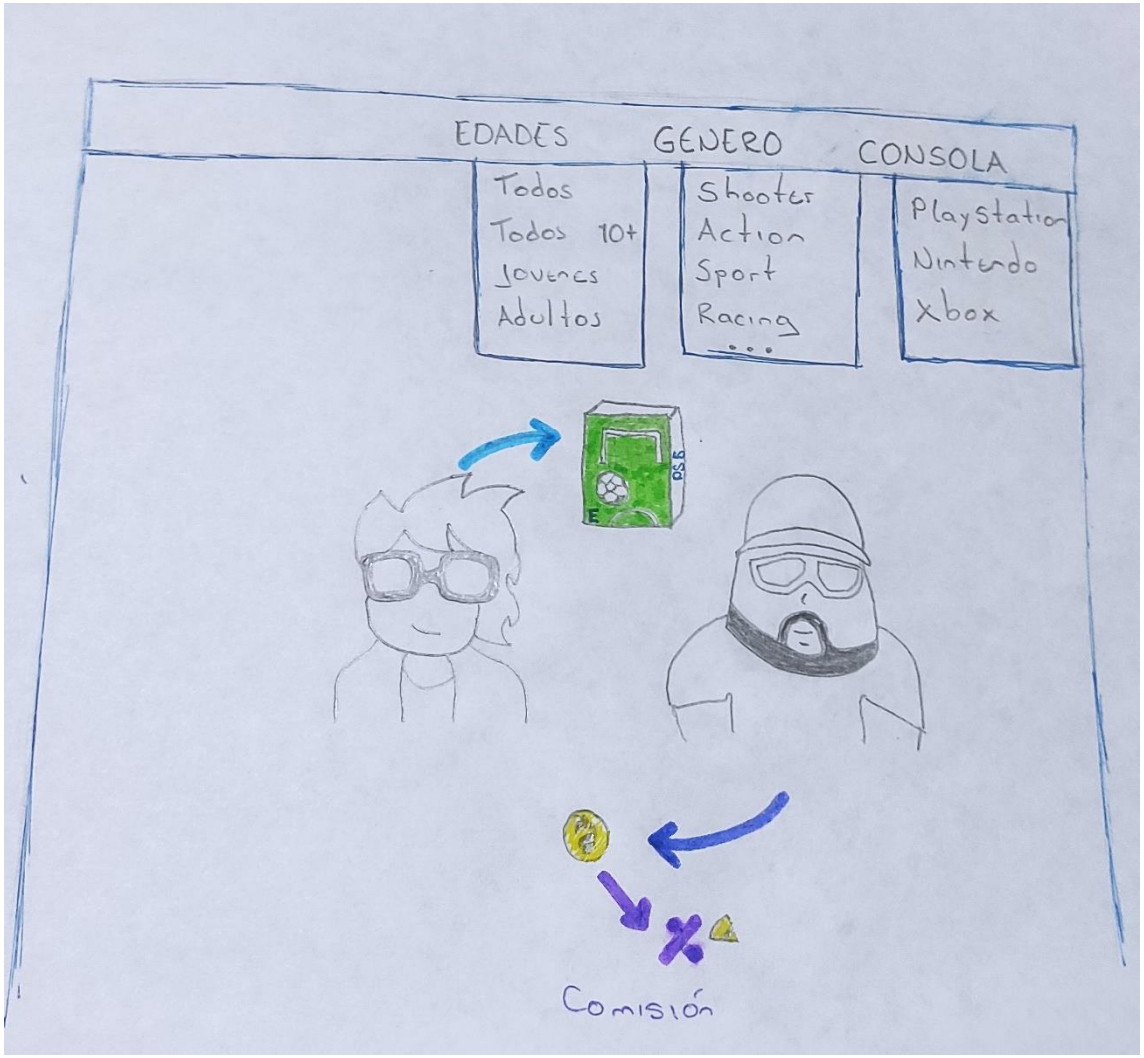
Fase 6. Despliegue (Puesta en producción)

Como simulación de despliegue se montará el código a GitHub, para que cualquier persona lo pueda consultar y mirar los resultados.

https://github.com/camilotobon18/videogames_innovacion/blob/master/video_games.ipynb

Design Thinking

Se toma en consideración la herramienta de prototipado, más específicamente la herramienta de prototipado en papel, para diseñar un aplicativo web de intercambio de video juegos, en el cual se cobra una comisión por el intercambio. La idea es diseñar una aplicativo que ponga en contacto vendedores y compradores de video juegos, ya que cuando un usuario culmino un video juego tiende a cambiarlo o a venderlo, con el fin de adquirir uno nuevo.


EDADES
Todos
Todos 10+
Jovencs
Adultos
GENERO
Shooter
Action
Sport
Racing
...
CONSOLA
Playstation
Nintendo
Xbox
PS5
E
Comisión