Tipología y ciclo de vida de los datos · Práctica 2
Francisco Enrique Rico Rocamora y Sergio Villalba Canales

Práctica 2: Limpieza y análisis de datos

El notebook correspondiente de esta práctica está alojado en: https://github.com/svillalc/UOC_Tipologia_Prac2

En esta práctica se realizará el tratamiento del dataset de Kaggle "Video Game Sales with Ratings", que contiene un ranking de videojuegos más vendidos de la historia. Se puede consultar en la siguiente URL: https://www.kaggle.com/rush4ratio/video-game-sales-with-ratings

1. Descripción del dataset

Nuestro dataset tiene las siguientes caracteristicas:

- Name: Nombre del videojuego.
- Platform: Platamorfa en la que ha sido lanzada el videojuego.
- Year_of_release: Año de lanzamiento.
- Genre: Género al quie pertenece el registro.
- Publisher: Empresa lanzadora del juego.
- NA_sales: Numero de ventas en Norte América en millones de unidades.
- EU_sales: Numero de ventas en Europa en millones de unidades.
- JP_sales: Numero de ventas en Japón en millones de unidades.
- Other_sales: Numero de ventas en las regiones no indicadas en los atributos anteriores en millones de unidades.
- Global_sales: Numero de ventas totales en millones de unidades.

El dataset busca recopilar datos acerca de los videojuegos más vendidos de la historia. Contiene campos descriptivos del videojuego, como pueden ser el nombre o el desarrollador; datos de ventas por región y globales; y puntuaciones de los críticos y de los usuarios.

En esta práctica buscaremos encontrar la posible relación existente entre algunas de las variables. Por ejemplo, podríamos establecer un modelo que a partir de las puntuaciones de usuarios y crítica, trate de predecir las ventas que tendrá el juego. También plantearemos hipótesis para contrastar grupos de datos, como por ejemplo diferencias en los datos de ventas entre dos desarrolladores.

2. Integración y selección de los datos de interés a analizar

In [1]:

```
%matplotlib inline
# Importamos el dataset, descargado directamente desde Kaggle
import pandas as pd
pd.options.mode.chained_assignment = None  # default='warn'
df = pd.read_csv('Video_Games_Sales_as_at_22_Dec_2016.csv')
```

In [2]:

```
# Mostramos las primeras filas del dataset
df.head()
```

Out[2]:

	Name	Platform	Year_of_Release	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales	Critic_Sco
0	Wii Sports	Wii	2006.0	Sports	Nintendo	41.36	28.96	3.77	8.45	82.53	76
1	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24	Na
2	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.68	12.76	3.79	3.29	35.52	82
3	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.61	10.93	3.28	2.95	32.77	80
4	Pokemon Red/Pokemon	GB	1996.0	Role-	Nintendo	11.27	8.89	10.22	1.00	31.37	Na

	Name	Platform	Year_of_Release	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales	Critic_Sco
	Rune Factory: A Fantasy Harvest Moon	DS	2009.0	Playing	Nintendo	0.11	0.07	0.22	0.03	0.43	

Vemos que existen bastantes registros que no contienen la información acerca de las puntuaciones, por lo que estos datos serán descartados puesto que no son válidos para nuestros intereses. Podríamos buscar un método para imputarlos, pero buscamos obtener un modelo de precisión por lo que preferimos tratar datos reales. Además, el dataset resultante de eliminar los registros sin críticas sigue siendo bastante grande, como veremos a continuación.

In [3]:

```
df.shape[0]
```

Out[3]:

```
16719
```

In [4]:

```
df[~df['Critic_Score'].isna()].shape[0]
```

Out[4]:

```
8137
```

Así, nuestro dataset será el resultante de eliminar los registros sin información acerca de las puntuaciones.

In [5]:

```
df = df[~df['Critic_Score'].isna()]
```

In [6]:

```
df.head()
```

Out[6]:

	Name	Platform	Year_of_Release	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales	Critic_Score	Cri
0	Wii Sports	Wii	2006.0	Sports	Nintendo	41.36	28.96	3.77	8.45	82.53	76.0	
2	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.68	12.76	3.79	3.29	35.52	82.0	
3	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.61	10.93	3.28	2.95	32.77	80.0	
6	New Super Mario Bros.	DS	2006.0	Platform	Nintendo	11.28	9.14	6.50	2.88	29.80	89.0	
7	Wii Play	Wii	2006.0	Misc	Nintendo	13.96	9.18	2.93	2.84	28.92	58.0	

Ahora nos quedaremos con las variables que nos interesan.

In [7]:

```
columns = ['Name', 'Platform', 'Genre', 'Publisher', 'Global_Sales', 'Critic_Score', 'User_Score']
df = df[columns]
```

3. Limpieza de los datos

Vemos los tipos de nuestras columnas. Nos interesa especialmente que los campos de ventas y puntuaciones sean campos numéricos.

In [8]:

```
df.dtypes
```

Out[8]:

```
Name             object
Platform         object
Genre            object
Publisher        object
Global_Sales    float64
Critic_Score    float64
User_Score       object
dtype: object
```

Al intentar transformar la columna "User_Score" a tipo numérico, obtenemos un error debido a que algunos valores registros informan este campo con el valor "tbd". Los eliminaremos también del dataset

In [9]:

```
df["User_Score"] = pd.to_numeric(df["User_Score"])
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
pandas/_libs/lib.pyx in pandas._libs.lib.maybe_convert_numeric()

ValueError: Unable to parse string "tbd"

During handling of the above exception, another exception occurred:

ValueError                                Traceback (most recent call last)
<ipython-input-9-4b6981828eea> in <module>
----> 1 df["User_Score"] = pd.to_numeric(df["User_Score"])

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-
packages/pandas/core/tools/numeric.py in to_numeric(arg, errors, downcast)
    149             coerce_numeric = errors not in ("ignore", "raise")
    150             values = lib.maybe_convert_numeric(
--> 151                 values, set(), coerce_numeric=coerce_numeric
    152             )
    153

pandas/_libs/lib.pyx in pandas._libs.lib.maybe_convert_numeric()

ValueError: Unable to parse string "tbd" at position 207
```

In [10]:

```
df = df[df["User_Score"] != 'tbd']
df["User_Score"] = pd.to_numeric(df["User_Score"])
df.dtypes
```

Out[10]:

```
Name             object
Platform         object
Genre            object
Publisher        object
Global_Sales    float64
Critic_Score    float64
User_Score      float64
dtype: object
```

Ahora vemos los valores perdidos por columnas:

In [11]:

```
df.isna().sum()
```

Out[11]:

```
Name             0
Platform         0
Genre            0
Publisher        4
Global_Sales     0
Critic_Score     0
User_Score      38
dtype: int64
```

Seguimos teniendo valores perdidos en los campos de puntuaciones, en concreto en el "User_Score". Eliminamos estos registros. Respecto a los valores perdidos en el campo "Publisher", los sustituiremos por una etiqueta por defecto, como por ejemplo "Unknown", debido a que se trata de un campo categórico.

In [12]:

```
df = df[~df['User_Score'].isna()]
df["Publisher"].fillna("Unknown", inplace=True)
df.isna().sum()
```

Out[12]:

```
Name             0
Platform         0
Genre            0
Publisher        0
Global_Sales     0
Critic_Score     0
User_Score       0
dtype: int64
```

Ahora representaremos un boxplot para cada una de nuestras 3 variables numéricas, tratando de indentificar así los outliers. En primer lugar veremos un breve resumen de las características de estas variables.

In [13]:

```
df.describe()
```

Out[13]:

	Global_Sales	Critic_Score	User_Score
count	7017.000000	7017.000000	7017.000000
mean	0.767049	70.249822	7.182428
std	1.940317	13.880646	1.441241
min	0.010000	13.000000	0.500000
25%	0.110000	62.000000	6.500000
50%	0.290000	72.000000	7.500000
75%	0.750000	80.000000	8.200000
max	82.530000	98.000000	9.600000

Dividimos el atributo "Critic_Score" entre 10 para situarnos en la misma escala que el atributo User_Score, es decir, medir la puntuación en una escala de 0 a 10.

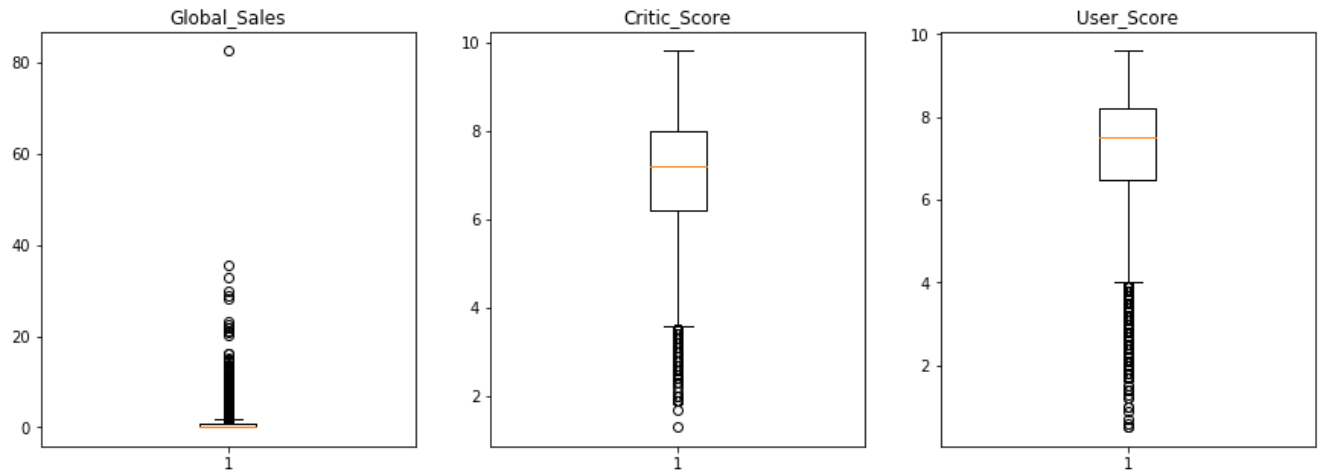
In [14]:

```
df["Critic_Score"] = df["Critic_Score"]/10
```

In [15]:

```
import matplotlib.pyplot as plt
fig, (ax0, ax1, ax2) = plt.subplots(ncols=3, figsize = (15, 5))
ax0.set_title('Global_Sales')
ax0.boxplot(df["Global_Sales"])
ax1.set_title('Critic_Score')
ax1.boxplot(df["Critic_Score"])
```

```
ax2.set_title('User_Score')
ax2.boxplot(df["User_Score"])
plt.show()
```



Vemos una gran cantidad de outliers en cada variable.

- Los outliers en la variable de ventas globales se deben a que la media de ventas de los videojuegos oscila en valores bastante bajos (la media era 0,76 millones), pero los juegos que ocupan la parte superior del ranking tienen una gran cantidad de ventas, de varios millones. Deberíamos quitar estos valores puesto que pueden perjudicar la calidad del modelo de predicción de ventas. Consideramos que el hecho de que un juego sea un superventas es algo puntual y nuestro modelo no lo valorará y predecirá ventas que ronden valores más bajos y habituales.
- Los outliers en las variables de puntuación se deben a puntuaciones muy por debajo de la media, que suele estar sobre el 7/10. Estos valores sí que los mantendremos puesto que podrían ayudar al modelo a predecir unas ventas muy pobres cuando las críticas son muy malas.

In [16]:

```
import numpy as np
# Eliminamos los registros cuyas ventan están más alla de tres desviaciones típicas de la media
df = df[np.abs(df['Global_Sales']-df['Global_Sales'].mean()) <= (3*df['Global_Sales'].std())]
```

4. Análisis de los datos

Realizaremos tres análisis distintos e independientes sobre nuestro dataset.

4.1. Regresión

En primer lugar, vamos a tratar de predecir las ventas que tendrá un videojuego a partir de las puntuaciones de la crítica y los usuarios

In [17]:

```
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score
X = np.array(df[["Critic_Score", "User_Score"]])
y = df['Global_Sales'].values

regr = linear_model.LinearRegression()

regr.fit(X, y)

y_pred = regr.predict(X)

print('Coefficients: \n', regr.coef_)
print('Independent term: \n', regr.intercept_)
print("Mean squared error: %.2f" % mean_squared_error(y, y_pred))
print('Variance score: %.2f' % r2_score(y, y_pred))
```

```
Coefficients:
 [ 0.26037735 -0.05477589]
Independent term:
 -0.809480546340199
Mean squared error: 0.72
Variance score: 0.12
```

El coeficiente R2 obtenido ha sido de 0.12, por lo que este modelo es bastante pobre y no ha ajustado correctamente el dataset. Esto nos permite confirmar que no existe una clara relación entre las puntuaciones y las ventas que tendrá un juego.

4.2. Clasificación

Buscamos una nueva forma de hallar la relación entre nuestras variables numéricas. En concreto, trataremos de predecir la puntuación de los usuarios de un videojuego a partir de las ventas y la puntuación de la crítica. Lo haremos mediante un algoritmo de clasificación, por lo que previamente discretizaremos los datos para que el algoritmo trate de predecir el grupo al que pertenecerá cada videojuego.

Creamos un nuevo dataframe con los ratings y ventas globales

In [18]:

```
columns = ['Global_Sales','Critic_Score', 'User_Score']
df2 = df[columns]
```

Discretizamos las variables en 11 Puntuaciones del 0 al 10.

In [19]:

```
df2["Critic_Score"]=round(df2["Critic_Score"])
df2["User_Score"]=round(df2["User_Score"])
```

Nuestras variables categorizadas tendrian la siguiente distribución de elementos:

In [20]:

```
df2.head()
```

Out[20]:

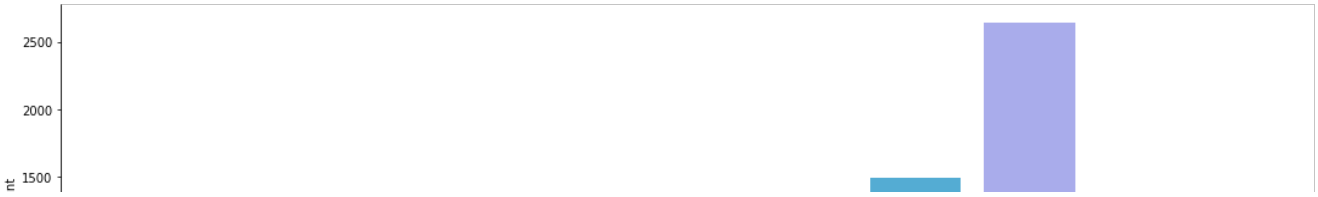
	Global_Sales	Critic_Score	User_Score
125	6.49	10.0	9.0
126	6.47	9.0	4.0
127	6.45	8.0	7.0
128	6.44	9.0	9.0
129	6.43	10.0	9.0

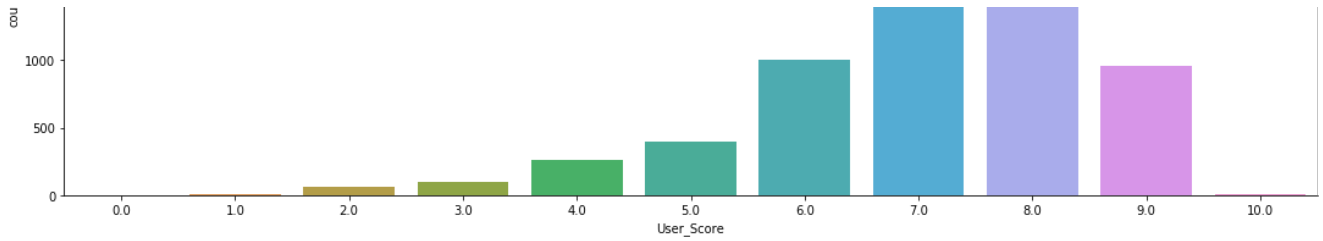
In [21]:

```
import seaborn as sb
sb.catplot('User_Score',data=df2,kind="count", aspect=3)
```

Out[21]:

```
<seaborn.axisgrid.FacetGrid at 0x121bba0b8>
```



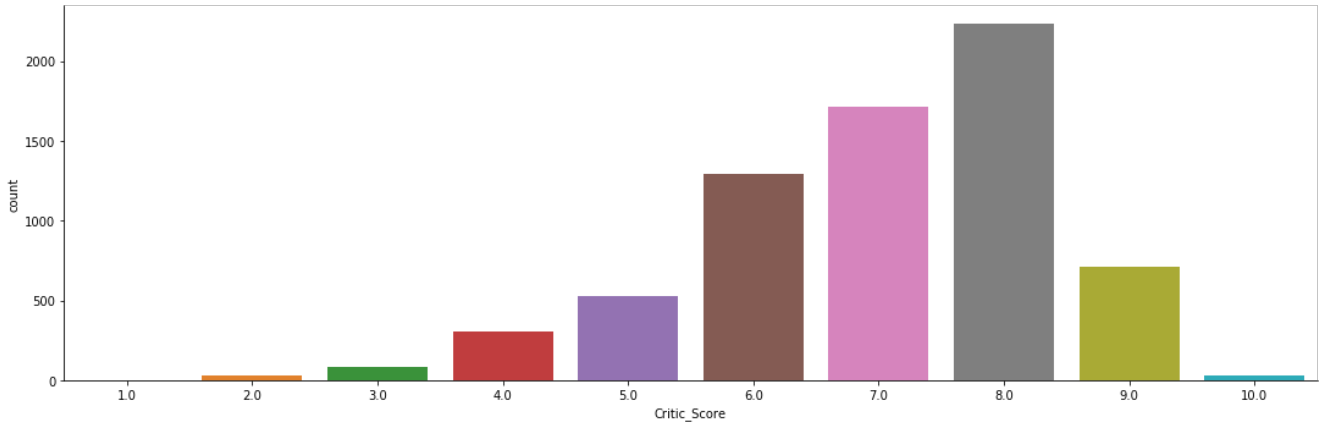


In [22]:

```
sb.catplot('Critic_Score',data=df2,kind="count", aspect=3)
```

Out[22]:

```
<seaborn.axisgrid.FacetGrid at 0x110e3b780>
```

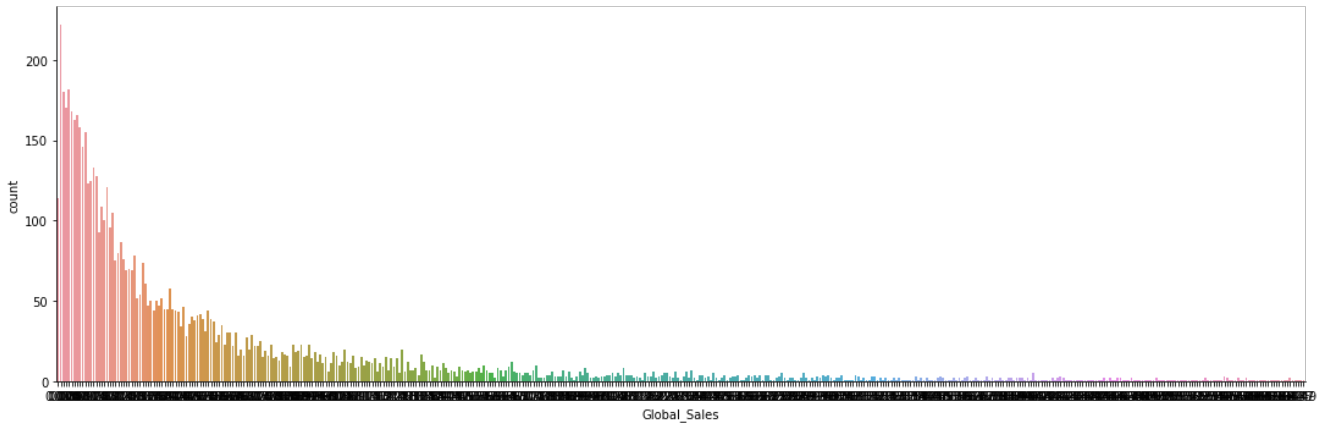


In [23]:

```
sb.catplot('Global_Sales',data=df2,kind="count", aspect=3)
```

Out[23]:

```
<seaborn.axisgrid.FacetGrid at 0x121f2d320>
```



Realicemos un algoritmo de clasificación a través de K-nearest neightbors. Utilizaremos 11 vecinos ya que son las clasificaciones que queremos obtener.

In [24]:

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
X = df2[['Global_Sales', 'Critic_Score']].values
y = df2['User_Score'].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,random_state=0)
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
```

```
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [25]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
n_neighbors = 11

knn = KNeighborsClassifier(11)
knn.fit(X_train, y_train)
print('Accuracy of K-NN classifier on training set: {:.2f}'
     .format(knn.score(X_train, y_train)))
print('Accuracy of K-NN classifier on test set: {:.2f}'
     .format(knn.score(X_test, y_test)))

predicted = knn.predict(X_test)
accuracy_score(y_test, predicted)
```

```
Accuracy of K-NN classifier on training set: 0.45
Accuracy of K-NN classifier on test set: 0.38
```

Out[25]:

```
0.3791446419990389
```

Obtenemos una precisión del 38% lo cual significa que nuestro algoritmo no ajusta demasiado bien a los datos que tenemos.

Apliquemos ahora el algoritmo Naive Bayes.

In [26]:

```
from sklearn.naive_bayes import GaussianNB
clf = GaussianNB()
clf.fit(X_train, y_train)


accuracy_score(y_test, clf.predict(X_test))
```

Out[26]:

```
0.2719846227775108
```

Vemos que la precisión baja al 27% por lo que este modelo clasifica peor que el anterior.

Por último veamos mediante el algoritmo Random Forest.

In [27]:

```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_estimators = 1000, random_state = 42)
rf.fit(X_train, y_train)
pred= rf.predict(X_test)

accuracy_score(y_test, np.round(pred))
```

Out[27]:

```
0.3695338779432965
```

Obtenemos una precisión cercana al 37% por lo que se aproxima al resultado del algoritmo Knn.

4.3. Constraste de hipótesis

En este apartado, realizaremos un contraste de hipótesis para estudiar la diferencia de medias de puntuaciones y ventas entre dos desarrolladores de videojuegos distintos. Empecemos viendo los desarrolladores de los cuales tenemos más registros en nuestro

conjunto de datos:

In [28]:

```
df.groupby('Publisher').count().sort_values(by=['Name']).tail()
```

Out[28]:

Publisher	Name	Platform	Genre	Global_Sales	Critic_Score	User_Score
Sony Computer Entertainment	305	305	305	305	305	305
THQ	309	309	309	309	309	309
Activision	484	484	484	484	484	484
Ubisoft	496	496	496	496	496	496
Electronic Arts	948	948	948	948	948	948

Realizaremos el contraste seleccionando los registros de las compañías Activision y Ubisoft, puesto que tenemos un número similar de registros de cada una y además es una muestra bastante grande (casi 500).

In [29]:

```
df_Activision = df[df['Publisher'] == 'Activision']
df_Ubisoft = df[df['Publisher'] == 'Ubisoft']
```

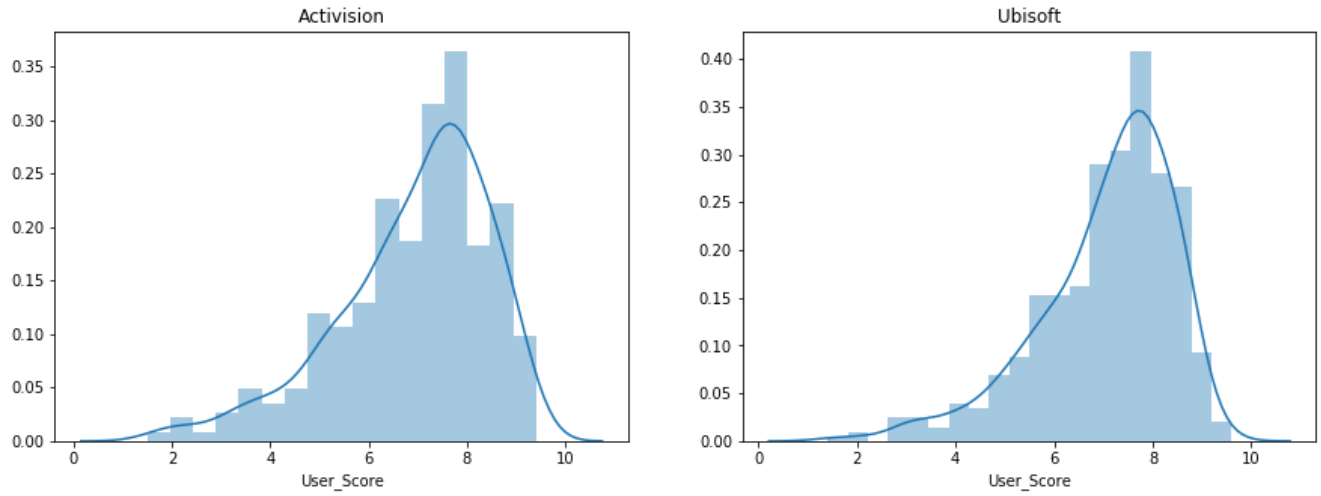
Vamos a visualizar la distribución de cada una de las variables numéricas de nuestros dos nuevos datasets filtrados. Realizamos una comparación visual de la distribición de cada variable en ambos datasets:

In [30]:

```
fig, axs = plt.subplots(ncols=2, figsize=(15,5))
sb.distplot(df_Activision['User_Score'], ax=axs[0]).set_title('Activision')
sb.distplot(df_Ubisoft['User_Score'], ax=axs[1]).set_title('Ubisoft')
```

Out[30]:

```
Text(0.5, 1.0, 'Ubisoft')
```

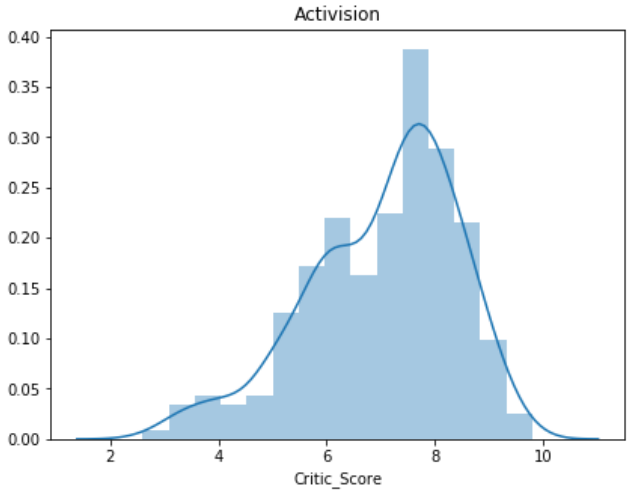


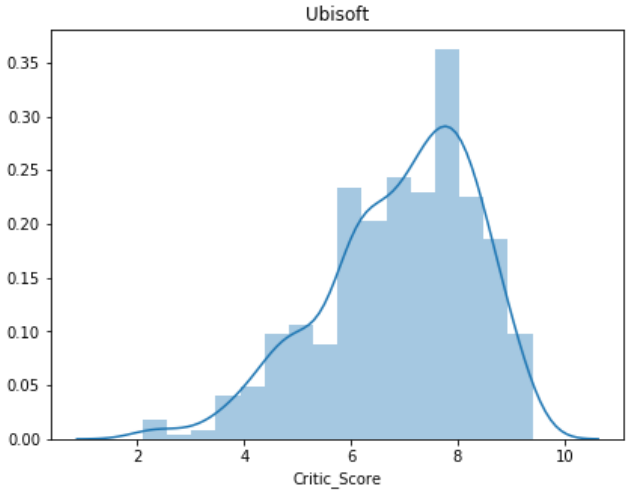
In [31]:

```
fig, axs = plt.subplots(ncols=2, figsize=(15,5))
sb.distplot(df_Activision['Critic_Score'], ax=axs[0]).set_title('Activision')
sb.distplot(df_Ubisoft['Critic_Score'], ax=axs[1]).set_title('Ubisoft')
```

Out[31]:

```
Text(0.5, 1.0, 'Ubisoft')
```



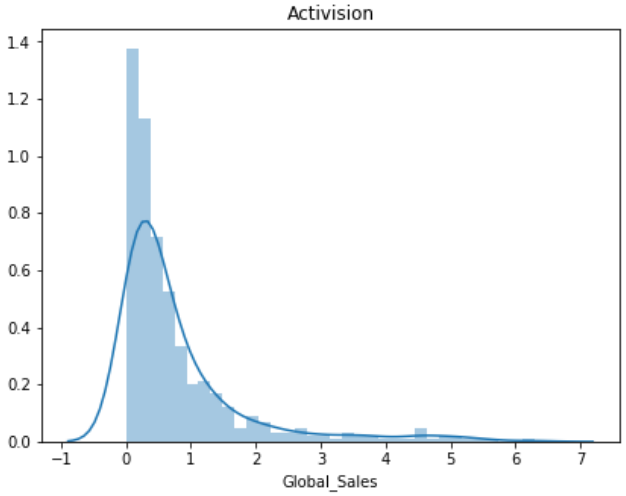


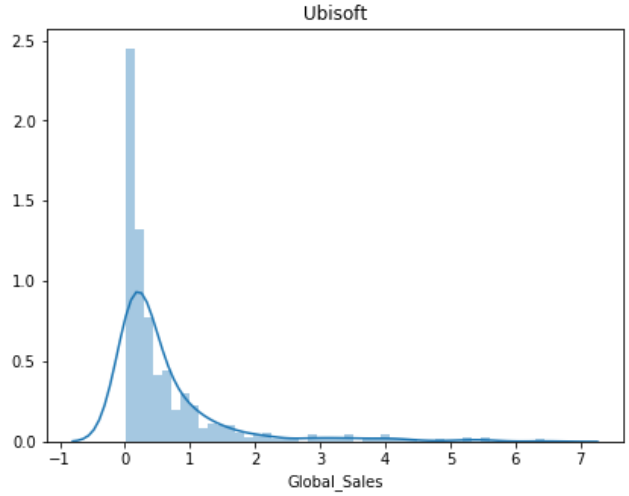
In [32]:

```
fig, axs = plt.subplots(ncols=2, figsize=(15,5))
sb.distplot(df_Activision['Global_Sales'], ax=axs[0]).set_title('Activision')
sb.distplot(df_Ubisoft['Global_Sales'], ax=axs[1]).set_title('Ubisoft')
```

Out[32]:

```
Text(0.5, 1.0, 'Ubisoft')
```





Las gráficas muestran que cada variable sigue una distribución similar en cada dataset, y que en ningún caso parece ser una distribución normal. Las variables de puntuación muestran un sesgo hacia la derecha, mientras que la de ventas muestra un sesgo hacia la izquierda.

Comprobemos la normalidad de cada variable en cada dataset mediante el test de Shapiro-Wilk:

In [33]:

```
from scipy.stats import shapiro
stat, p = shapiro(df_Activision['User_Score'])
print("p-valor para la hipótesis nula de normalidad en el User_Score de Activision: " + str(p))
stat, p = shapiro(df_Ubisoft['User_Score'])
print("p-valor para la hipótesis nula de normalidad en el User_Score de Ubisoft: " + str(p))
stat, p = shapiro(df_Activision['Critic_Score'])
print("p-valor para la hipótesis nula de normalidad en el Critic_Score de Activision: " + str(p))
stat, p = shapiro(df_Ubisoft['Critic_Score'])
print("p-valor para la hipótesis nula de normalidad en el Critic_Score de Ubisoft: " + str(p))
stat, p = shapiro(df_Activision['Global_Sales'])
print("p-valor para la hipótesis nula de normalidad en el Global_Sales de Activision: " + str(p))
stat, p = shapiro(df_Ubisoft['Global_Sales'])
print("p-valor para la hipótesis nula de normalidad en el Global_Sales de Ubisoft: " + str(p))
```

```
p-valor para la hipótesis nula de normalidad en el User_Score de Activision: 1.9444477495147994e-1
```

```
3
p-valor para la hipótesis nula de normalidad en el User_Score de Ubisoft: 4.3197375879230546e-15
p-valor para la hipótesis nula de normalidad en el Critic_Score de Activision: 1.196673093950551e-09
p-valor para la hipótesis nula de normalidad en el Critic_Score de Ubisoft: 3.159573436217755e-10
p-valor para la hipótesis nula de normalidad en el Global_Sales de Activision: 2.248094088806364e-29
p-valor para la hipótesis nula de normalidad en el Global_Sales de Ubisoft: 8.353230952635343e-32
```

Tal y como deducíamos a partir de las gráficas, en todos los casos rechazamos la hipótesis nula con un alto nivel de confianza

Ahora procedemos a realizar el contraste de hipótesis para la diferencia de medias de estas tres variables en nuestros dos datasets. Dado que tenemos dos muestras de más de 30 elementos, podemos aplicar el teorema central del límite y considerar que nuestras poblaciones son normales a la hora de realizar los tests. Por tanto, aplicamos test de Student.

In [34]:

```
from scipy.stats import ttest_ind
# User_Score
ttest_ind(df_Ubisoft['User_Score'], df_Activision['User_Score'])
```

Out[34]:

```
Ttest_indResult(statistic=1.4985084863934994, pvalue=0.13432388039854457)
```

In [35]:

```
# Critic_Score
ttest_ind(df_Ubisoft['Critic_Score'], df_Activision['Critic_Score'])
```

Out[35]:

```
Ttest_indResult(statistic=-1.0185667307043118, pvalue=0.30866060412453405)
```

In [36]:

```
# Global_Sales
ttest_ind(df_Ubisoft['Global_Sales'], df_Activision['Global_Sales'])
```

Out[36]:

```
Ttest_indResult(statistic=-2.7166875986126824, pvalue=0.00671027801900192)
```

A partir de los resultados de los tests, podemos confirmar que con un 95% de confianza, no podemos asegurar la diferencia de medias entre las puntuaciones de usuarios y crítica de Activision y Ubisoft. Sin embargo, sí que podemos confirmar la diferencia de medias en las ventas. Veamos sus respectivas medias:

In [37]:

```
print(df_Ubisoft['Global_Sales'].mean())
print(df_Activision['Global_Sales'].mean())
```

```
0.620625
0.7918595041322315
```

La media de ventas de los juegos de Activision es considerablemente mayor. Este contraste nos ha permitido ver que, aunque dos compañías publiquen juegos de similar calidad, teniendo en cuenta las puntuaciones, las diferencias de ventas pueden ser realmente grandes. Esto nos permite deducir que para los desarrolladores, existen muchos más aspectos a tener en cuenta a la hora de publicar un videojuego aparte de su calidad, como por ejemplo la publicidad, puesto que pueden incrementar en gran volumen las ventas.

Este contraste también reafirma de alguna manera lo que hemos experimentado en los dos modelos anteriores: es difícil ajustar las ventas a partir de las puntuaciones, o vicecersa, puesto que se nos escapan muchos aspectos que no estamos teniendo en cuenta. Por ello obteníamos precisiones tan bajas.

5. Representación de los resultados a partir de tablas y gráficas

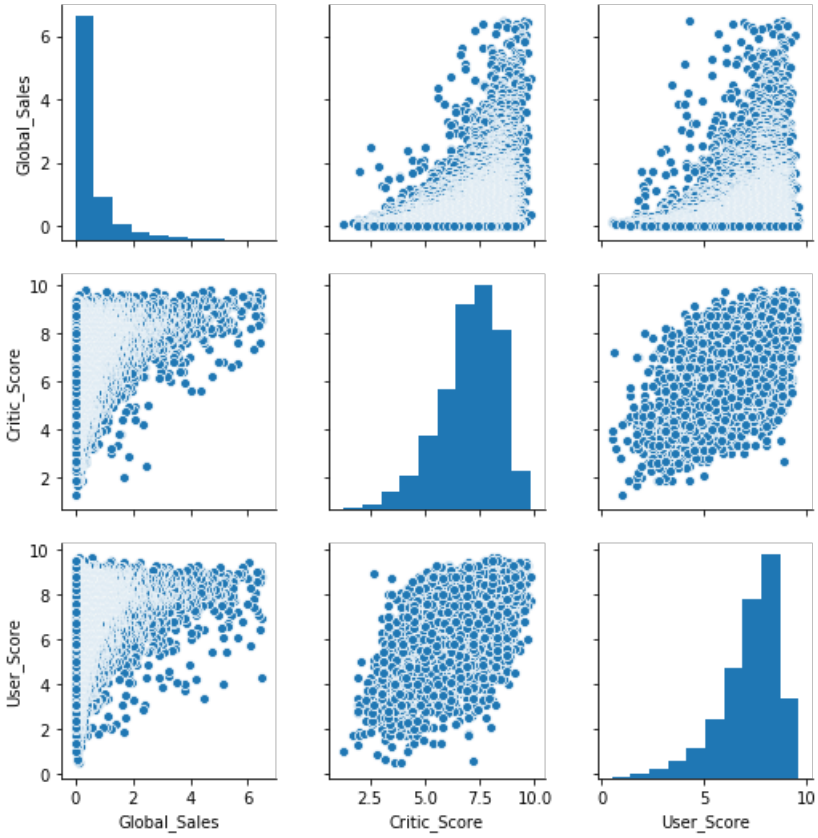
Durante los apartados anteriores ya hemos ido representando gráficas que apoyaban los análisis realizados. En este apartado aportaremos nuevas gráficas que ilustrarán los resultados con otro enfoque. Por ejemplo, empezamos representando un pairplot con las tres variables numéricas de nuestro dataset. Así podemos apreciar de nuevo la dificultad de establecer un modelo de regresión lineal entre ellas.

In [38]:

```
sb.pairplot(df)
```

Out[38]:

```
<seaborn.axisgrid.PairGrid at 0x123b872e8>
```



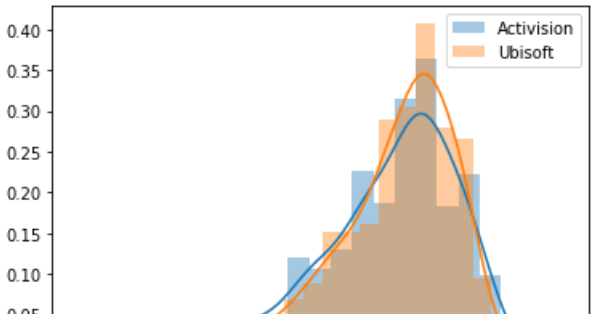
También vamos a representar la distribución de las tres variables numéricas sobre los conjuntos de datos de Ubisoft y Activision, pero representando ambos conjuntos sobre la misma gráfica para ayudar a comparar.

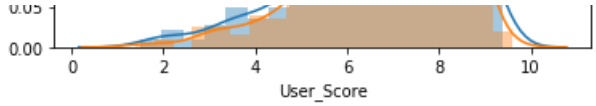
In [39]:

```
sb.distplot(df_Activision['User_Score'], label="Activision")
sb.distplot(df_Ubisoft['User_Score'], label = "Ubisoft")
plt.legend()
```

Out[39]:

```
<matplotlib.legend.Legend at 0x1240939e8>
```



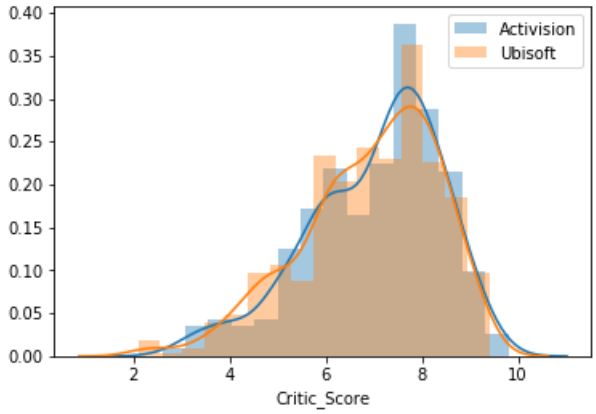


In [40]:

```
sb.distplot(df_Activision['Critic_Score'], label="Activision")
sb.distplot(df_Ubisoft['Critic_Score'], label="Ubisoft")
plt.legend()
```

Out[40]:

```
<matplotlib.legend.Legend at 0x124112080>
```

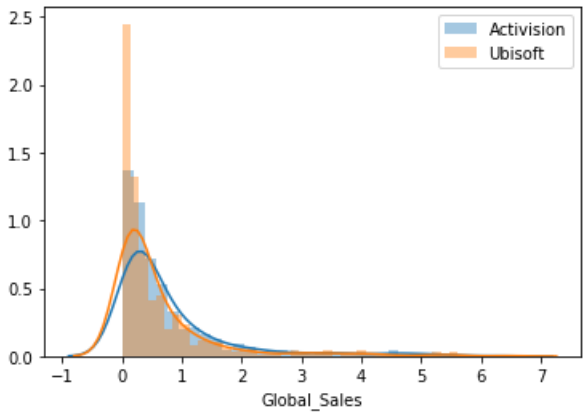


In [41]:

```
sb.distplot(df_Activision['Global_Sales'], label="Activision")
sb.distplot(df_Ubisoft['Global_Sales'], label="Ubisoft")
plt.legend()
```

Out[41]:

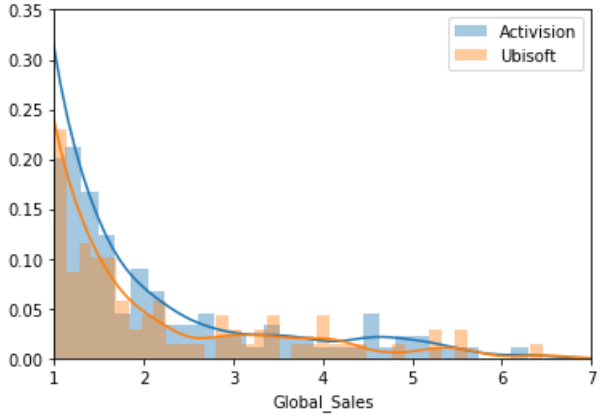
```
<matplotlib.legend.Legend at 0x12449c5c0>
```



Vemos la similitud de las gráficas para las variables de puntuaciones. Sin embargo, priori es difícil apreciar la diferencia en las ventas a partir del último gráfico. Si hacemos zoom sobre la "cola" de la gráfica, es decir, sobre los videojuegos correspondientes a superventas, sí que podemos apreciar que las frecuencias son mayores en el dataset de Activision.

In [42]:

```
fig, ax0 = plt.subplots()
sb.distplot(df_Activision['Global_Sales'], ax=ax0, label = "Activision")
sb.distplot(df_Ubisoft['Global_Sales'], ax=ax0, label = "Ubisoft")
ax0.set_xlim([1, 7])
ax0.set_ylim([0, 0.35])
plt.legend()
plt.show()
```



6. Conclusiones

A rasgos generales nos damos cuentas que el dataset como concepto es interesante para el estudio, pero en la práctica no es tan bueno debido a que no podemos extraer conclusiones precisas de los métodos y recursos aplicados.

En el apartado 4 hemos aplicado tres métodos distintos de analisis a nuestros datos. En primer lugar, hemos aplicado una regresión lineal con el objetivo de predecir el número de ventas en función de la nota obtenida por los críticos y los usuarios. Dicha regresión nos ha devuelto un coeficiente de determinación de 0,12 lo cual nos indica que nuestro modelo no ajusta correctamente ya que dicho coeficiente está alejado de 1.

Seguidamente, hemos implementados algoritmos de clasificación con el objetivo de predecir nuevamente valores futuros en función de las mismas variables que en la regresión lineal. Los métodos empleados han sido K- vecinos más cercanos, Naive Bayes, y arbol de decisión aleatorio, obteniendo una precisión 38%, 27% y 37% respectivamente. Este resultado nos indica que nuestros algoritmos no realizan una buena clasificación y por ello no no deben ser usados para predecir grupos teniendo el resto de variables.

Finalmente, en este bloque hemos realizado un contraste de hipótesis sobre las diferencias de medias de ventas y puntuación obtenida en los videojuegos para dos grupos de desarrolladores, Activision y Ubisoft. En dichos contrastes, hemos obtenido que no podemos afirmas diferencias entre las medias de las puntuaciones ya que el p-valor obtenido es mayor que 0.05. Sin embargo,si que podemos afirmas que e las ventas de Activision superaran en media a las de Ubisoft.

Por último, para finalizar el trabajo, hemos realizado una serie de gráficas que nos permiten afianzar más los resultados obtenidos en los apartados anteriores.

In [ ]: